TC1130

32-Bit Single-Chip Microcontroller

Volume 2 (of 2): Peripheral Units

Microcontrollers

Never stop thinking.

Edition 2004-11

Published by Infineon Technologies AG,
St.-Martin-Strasse 53,
81669 München, Germany

© Infineon Technologies AG 2005.
All Rights Reserved.

Attention please!

The information herein is given to describe certain components and shall not be considered as a guarantee of characteristics.

Terms of delivery and rights to technical change reserved.

We hereby disclaim any and all warranties, including but not limited to warranties of non-infringement, regarding circuits, descriptions and charts stated herein.

Information

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies Office.

Infineon Technologies Components may only be used in life-support devices or systems with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system, or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

TC1130

32-Bit Single-Chip Microcontroller

Volume 2 (of 2): Peripheral Units

Microcontrollers

Never stop thinking.

TC1130	**Volume 2 (of 2): Peripheral Units**
Revision History:	**V1.3, 2004-11**
Previous Version:	V1.2, 2004-07 V1.1, 2004-05 V1.0, 2004-03

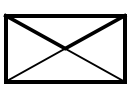
Page	Subjects (major changes since last revision)
24-23	Note for “specific slave select output mode” is added.
25-6	“SYSCON” is changed to “BUSCON” in the first paragraph of Section 25.1.3.3.
25-22	Switched bit field names of “CLRWMEN” and “SETWMEN” in field column.
26-29	Switched bit field names of “USBSRL” and “CPUSRL” in field column.
26-30	Description of register bit “UCLK” is corrected.
27-63	Description of register bit field TCR.MNAE is updated.
27-68	Description of register bit field TCMDR.CMDP0 is corrected.
27-70	Description of register bit RSTATR.RPx is updated.
27-73, 27-78	Description of BS bit field is corrected.
27-75	Description of register bit RCR.TF is updated.
27-77	Description of register bit field RPxBAR.ADDR is corrected.
27-80	Description of register bit SCR.SCVx and SCR.CMOD are updated.
27-107	Switched bit field names of “SC” and “DM” in field column.
30-64	Bit 22 of NFCR register is corrected.

Controller Area Network (CAN): License of Robert Bosch GmbH

We Listen to Your Comments

If you feel any information in this document is unclear, incomplete or incorrect, please do let us know. Your feedback will help us to continuously improve the quality of this document.

Please send your comments (including a reference to this document) to:



mcdocu.comments@infineon.com

Table of Contents

This User's Manual consists of two volumes, "System Units" [1] and "Peripheral Units" [2]. For your convenience, this table of contents (and also the keyword index and the register index) lists both volumes, so you can immediately find the reference to the desired section in the corresponding document ([1] or [2]).

Table of Contents . . . . . L-1 [1+2]

1 Introduction . . . . . 1-1 [1]
1.1 About this Document . . . . . 1-1 [1]
1.1.1 Related Documentations . . . . . 1-1 [1]
1.1.2 Textual Conventions . . . . . 1-1 [1]
1.1.3 Reserved, Undefined and Unimplemented Terminology . . . . . 1-3 [1]
1.1.4 Register Access Modes . . . . . 1-3 [1]
1.1.5 Acronyms . . . . . 1-4 [1]
1.2 System Architecture Features of the TC1130 . . . . . 1-7 [1]
1.3 Block Diagram . . . . . 1-10 [1]
1.4 On-Chip Peripheral Units of the TC1130 . . . . . 1-11 [1]
1.4.1 Serial Interfaces . . . . . 1-12 [1]
1.4.1.1 Asynchronous/Synchronous Serial Interface (ASC) . . . . . 1-12 [1]
1.4.1.2 High-Speed Synchronous Serial Interface (SSC) . . . . . 1-15 [1]
1.4.1.3 Inter IC Serial Interface (IIC) . . . . . 1-17 [1]
1.4.1.4 Universal Serial Bus Interface (USB) . . . . . 1-19 [1]
1.4.1.5 Micro Link Serial Bus Interface (MLI) . . . . . 1-21 [1]
1.4.2 General Purpose Timer Unit . . . . . 1-23 [1]
1.4.3 Capture/Compare Unit 6 (CCU6) . . . . . 1-25 [1]
1.4.4 MultiCAN . . . . . 1-27 [1]
1.4.5 Ethernet Controller . . . . . 1-30 [1]
1.5 Pin Definitions and Functions . . . . . 1-32 [1]

2 TC1130 Processor Architecture . . . . . 2-1 [1]
2.1 Programming Model . . . . . 2-3 [1]
2.1.1 Architectural Registers . . . . . 2-3 [1]
2.1.2 Data Types . . . . . 2-4 [1]
2.1.3 Memory Model . . . . . 2-5 [1]
2.1.4 Addressing Modes . . . . . 2-6 [1]
2.2 Tasks and Contexts . . . . . 2-7 [1]
2.2.1 Context Save Area (CSA) . . . . . 2-8 [1]
2.2.2 Fast Context Switching . . . . . 2-8 [1]
2.3 Interrupt System . . . . . 2-9 [1]
2.3.1 Interrupt Priority . . . . . 2-9 [1]

Table of Contents

2.4 Trap System . . . 2-10 [1]
2.5 Memory Management Unit (MMU) . . . 2-10 [1]
2.6 Protection System . . . 2-11 [1]
2.7 Debug System . . . 2-12 [1]
2.8 Processor Registers . . . 2-12 [1]
2.8.1 General Purpose Registers (GPRs) . . . 2-14 [1]
2.8.2 Program State Information Registers . . . 2-15 [1]
2.8.2.1 Program Counter (PC) . . . 2-15 [1]
2.8.2.2 Program Status Word (PSW) . . . 2-17 [1]
2.8.2.3 Previous Context Information Register (PCXI) . . . 2-21 [1]
2.8.3 Context Management Registers . . . 2-23 [1]
2.8.3.1 Free Context List Head Pointer (FCX) . . . 2-23 [1]
2.8.3.2 Previous Context Pointer (PCX) . . . 2-24 [1]
2.8.4 Free Context List Limit Pointer (LCX) . . . 2-25 [1]
2.8.5 Stack Management . . . 2-26 [1]
2.8.5.1 Interrupt Stack Pointer (ISP) . . . 2-26 [1]
2.8.6 Interrupt and Trap Control . . . 2-27 [1]
2.8.6.1 Interrupt Control Register (ICR) . . . 2-27 [1]
2.8.6.2 Interrupt Vector Table Pointer (BIV) . . . 2-29 [1]
2.8.6.3 Trap Vector Table Pointer (BTV) . . . 2-30 [1]
2.8.7 System Control Register . . . 2-31 [1]
2.8.8 Memory Management Unit (MMU) Registers . . . 2-31 [1]
2.8.9 Memory Protection Registers . . . 2-32 [1]
2.8.10 Debug Registers . . . 2-32 [1]
2.9 Instruction Set Overview . . . 2-39 [1]
2.9.1 PSW Status Flags and Arithmetic . . . 2-39 [1]
2.9.2 Integer Arithmetic . . . 2-40 [1]
2.9.2.1 Move . . . 2-40 [1]
2.9.2.2 Addition and Subtraction . . . 2-40 [1]
2.9.2.3 Multiply and Multiply-Add . . . 2-41 [1]
2.9.2.4 Division . . . 2-41 [1]
2.9.2.5 Absolute Value, Absolute Difference . . . 2-42 [1]
2.9.2.6 Min, Max, Saturate . . . 2-42 [1]
2.9.2.7 Conditional Arithmetic Instructions . . . 2-42 [1]
2.9.2.8 Logic Operations . . . 2-43 [1]
2.9.2.9 Count Leading Zeros, Ones, and Signs . . . 2-43 [1]
2.9.2.10 Shift . . . 2-44 [1]
2.9.2.11 Bit Field Extract and Insert . . . 2-44 [1]
2.9.3 Packed Arithmetic . . . 2-47 [1]
2.9.4 DSP Arithmetic . . . 2-48 [1]
2.9.4.1 Scaling . . . 2-48 [1]
2.9.4.2 Special Case = -1 × -1 => +1 . . . 2-48 [1]
2.9.4.3 Guard Bits . . . 2-49 [1]

2.9.4.4 Rounding . . . 2-49 [1]
2.9.4.5 Overflow and Saturation . . . 2-49 [1]
2.9.4.6 Sticky Advance Overflow and Block Scaling in FFT . . . 2-49 [1]
2.9.4.7 Multiply and MAC . . . 2-49 [1]
2.9.4.8 Packed Multiply and Packed MAC . . . 2-50 [1]
2.9.5 Compare Instructions . . . 2-51 [1]
2.9.6 Bit Operations . . . 2-54 [1]
2.9.7 Address Arithmetic . . . 2-56 [1]
2.9.8 Address Comparison . . . 2-57 [1]
2.9.9 Branch Instructions . . . 2-58 [1]
2.9.9.1 Unconditional Branch . . . 2-58 [1]
2.9.9.2 Conditional Branch . . . 2-59 [1]
2.9.9.3 Loop Instructions . . . 2-60 [1]
2.9.10 Load and Store Instructions . . . 2-61 [1]
2.9.10.1 Load/Store Basic Data Types . . . 2-62 [1]
2.9.10.2 Load Bit . . . 2-63 [1]
2.9.10.3 Store Bit and Bit Field . . . 2-63 [1]
2.9.11 Context Related Instructions . . . 2-64 [1]
2.9.11.1 Context Saving and Restoring . . . 2-64 [1]
2.9.11.2 Context Loading and Storing . . . 2-65 [1]
2.9.12 System Instructions . . . 2-65 [1]
2.9.12.1 System Call . . . 2-65 [1]
2.9.12.2 Synchronization Primitives . . . 2-65 [1]
2.9.12.3 Access to the Core Special Function Registers (CSFRs) . . . 2-66 [1]
2.9.12.4 Enabling/Disabling the Interrupt System . . . 2-67 [1]
2.9.12.5 RET and RFE . . . 2-67 [1]
2.9.12.6 Trap Instructions . . . 2-67 [1]
2.9.12.7 No-operation (NOP) . . . 2-67 [1]
2.9.13 16-bit Instructions . . . 2-68 [1]
2.10 FPU . . . 2-69 [1]
2.10.1 Data Format . . . 2-69 [1]
2.10.2 Denormal Numbers . . . 2-69 [1]
2.10.3 Floating Point Registers . . . 2-70 [1]
2.10.4 Extended Precision . . . 2-70 [1]
2.10.5 Exception Conditions . . . 2-70 [1]
2.10.6 Rounding . . . 2-71 [1]
2.10.7 FPU Instructions . . . 2-72 [1]
2.10.7.1 Arithmetic Operations . . . 2-72 [1]
2.10.7.2 Non-Arithmetic Operations . . . 2-74 [1]
2.10.7.3 Conversion Operations . . . 2-74 [1]
2.10.8 Exception Handling . . . 2-76 [1]
2.10.8.1 Invalid Operation . . . 2-76 [1]
2.10.8.2 Divide by Zero . . . 2-77 [1]

2.10.8.3 Overflow . . . 2-77 [1]
2.10.8.4 Underflow . . . 2-77 [1]
2.10.8.5 Inexact . . . 2-78 [1]
2.10.9 Cycle Counts by Opcode . . . 2-78 [1]
2.11 CPU Slave Interface (CPS) . . . 2-80 [1]
2.11.1 Feature Summary . . . 2-80 [1]
2.11.2 SFRs of the CPU Slave Interface (CPS) . . . 2-80 [1]
2.12 CPU Register Address Ranges . . . 2-83 [1]

3 Clock System . . . 3-1 [1]
3.1 Clock Generation Unit . . . 3-3 [1]
3.2 Clock Registers . . . 3-5 [1]
3.2.1 Main Oscillator Circuit . . . 3-6 [1]
3.2.2 PLL Module . . . 3-10 [1]
3.2.2.1 PLL Functional Description . . . 3-11 [1]
3.2.2.2 PLL Clock Control and Status Register . . . 3-14 [1]
3.2.3 Clock Source Control . . . 3-17 [1]
3.2.3.1 Setting up the PLL after Reset . . . 3-18 [1]
3.2.3.2 Switching PLL Parameters . . . 3-18 [1]
3.2.4 Power-on Startup Operation . . . 3-19 [1]
3.2.5 Loss-of-Lock Operation . . . 3-19 [1]
3.2.6 Loss-of-Lock Recovery . . . 3-20 [1]
3.3 Module Power Management and Clock Gating . . . 3-21 [1]
3.3.1 Module Clock Generation . . . 3-22 [1]
3.3.1.1 Clock Control Registers . . . 3-23 [1]
3.3.1.2 Fractional Divider . . . 3-28 [1]
3.3.1.3 Module Clock Generation Implementations . . . 3-35 [1]

4 System Control Unit . . . 4-1 [1]
4.1 Overview . . . 4-1 [1]
4.2 Parity Error Control . . . 4-2 [1]
4.3 Faulty SRAM Fusebox . . . 4-6 [1]
4.4 CSCOMB (CSovl/CSglb) Control . . . 4-8 [1]
4.5 EBU Pull-Up Control . . . 4-8 [1]
4.6 DMA Request Signal Selection . . . 4-9 [1]
4.7 Miscellaneous SCU Registers . . . 4-12 [1]
4.8 SCU Registers and Address Map . . . 4-19 [1]
4.8.1 SCU Register Address Range . . . 4-22 [1]

5 Reset and Boot Operation . . . 5-1 [1]
5.1 Overview . . . 5-1 [1]
5.2 Reset Registers . . . 5-2 [1]
5.2.1 Reset Status Register (RST_SR) . . . 5-2 [1]
5.2.2 Reset Request Register (RST_REQ) . . . 5-4 [1]

5.3 Reset Operations . . . . . 5-6 [1]
5.3.1 Power-On Reset . . . . . 5-6 [1]
5.3.2 External Hard Reset . . . . . 5-6 [1]
5.3.3 Soft Reset . . . . . 5-7 [1]
5.3.4 Watchdog Timer Reset . . . . . 5-7 [1]
5.3.5 Deep Sleep Wake-Up Reset . . . . . 5-8 [1]
5.3.6 Debug System Reset . . . . . 5-9 [1]
5.3.7 State of the TC1130 After Reset . . . . . 5-9 [1]
5.4 Booting Scheme . . . . . 5-11 [1]
5.4.1 Boot Options . . . . . 5-11 [1]
5.4.2 Normal Boot Options . . . . . 5-12 [1]
5.4.3 Debug Boot Options . . . . . 5-13 [1]
5.5 Configuration Input Sampling . . . . . 5-14 [1]
5.5.1 Hardware Configuration Inputs . . . . . 5-14 [1]
5.5.2 Software Option Select Inputs . . . . . 5-14 [1]

6 Power Management . . . . . 6-1 [1]
6.1 Power Management Overview . . . . . 6-1 [1]
6.2 Power Management Control Registers . . . . . 6-3 [1]
6.2.1 Power Management Control Register PMG_CON . . . . . 6-3 [1]
6.2.2 Power Management Control and Status Register PMG_CSR . . . . . 6-5 [1]
6.3 Power Management Modes . . . . . 6-6 [1]
6.3.1 Idle Mode . . . . . 6-6 [1]
6.3.2 Sleep Mode . . . . . 6-6 [1]
6.3.2.1 Entering Sleep Mode . . . . . 6-6 [1]
6.3.2.2 TC1130 State During Sleep Mode . . . . . 6-7 [1]
6.3.2.3 Exiting Sleep Mode . . . . . 6-7 [1]
6.3.3 Deep Sleep Mode . . . . . 6-7 [1]
6.3.3.1 Entering Deep Sleep Mode . . . . . 6-8 [1]
6.3.3.2 TC1130 State During Deep Sleep Mode . . . . . 6-8 [1]
6.3.3.3 Exiting Deep Sleep Mode . . . . . 6-8 [1]
6.3.3.4 Exiting Deep Sleep Mode With a Power-On Reset Signal . . . . . 6-9 [1]
6.3.3.5 Exiting Deep Sleep Mode With an NMI Signal . . . . . 6-9 [1]
6.3.4 Summary of TC1130 Power Management States . . . . . 6-9 [1]

7 Memory Map of On-Chip Local Memories . . . . . 7-1 [1]

8 Program Memory Interface (PMI) . . . . . 8-1 [1]
8.1 Feature Summary and Block Diagram . . . . . 8-1 [1]
8.2 LMB Access Priorities . . . . . 8-2 [1]
8.3 Scratch-Pad RAM, SPRAM . . . . . 8-2 [1]
8.4 Instruction Cache, ICACHE . . . . . 8-4 [1]
8.4.1 Cache Organization . . . . . 8-4 [1]
8.4.2 Cache Bypass Control . . . . . 8-4 [1]

8.4.3 Refill Sequence for Cache . . . . . 8-4 [1]
8.4.4 Instruction Streaming . . . . . 8-5 [1]
8.4.5 Cache Coherency, Cache Invalidation . . . . . 8-5 [1]
8.5 PMI Registers . . . . . 8-6 [1]

9 Data Memory Interface (DMI) . . . . . 9-1 [1]
9.1 Feature Summary and Block Diagram . . . . . 9-1 [1]
9.2 LMB Access Priorities . . . . . 9-2 [1]
9.3 DMI Registers . . . . . 9-3 [1]

10 Memory Management Unit . . . . . 10-1 [1]
10.1 Address Spaces . . . . . 10-3 [1]
10.2 Address Translation . . . . . 10-4 [1]
10.2.1 Address Translation for Context Pointers . . . . . 10-4 [1]
10.3 Translation Lookaside Buffers (TLBs) . . . . . 10-5 [1]
10.3.1 TLB Table Entry Contents . . . . . 10-5 [1]
10.4 Cacheability . . . . . 10-6 [1]
10.4.1 Cacheability for Direct Translation . . . . . 10-6 [1]
10.4.2 Cacheability for PTE based Translation . . . . . 10-6 [1]
10.4.3 Complete Description . . . . . 10-6 [1]
10.5 Protection . . . . . 10-8 [1]
10.5.1 Protection for Direct Translation . . . . . 10-8 [1]
10.5.2 Protection for PTE Translation . . . . . 10-8 [1]
10.6 Multiple Address Spaces . . . . . 10-8 [1]
10.7 MMU Traps . . . . . 10-9 [1]
10.8 MMU Instructions . . . . . 10-11 [1]
10.8.1 TLBMAP (TLB Map) . . . . . 10-11 [1]
10.8.2 TLBDEMAP (TLB Demap) . . . . . 10-11 [1]
10.8.3 TLBFLUSH (TLB Flush) . . . . . 10-11 [1]
10.8.4 TLBPROBE (TLB Probe) . . . . . 10-12 [1]
10.9 MMU Registers . . . . . 10-12 [1]
10.9.1 Configuration Register . . . . . 10-13 [1]
10.9.2 Address Space Identifier Register . . . . . 10-15 [1]
10.9.3 Translation Virtual Address Register . . . . . 10-15 [1]
10.9.4 Translation Physical Address Register . . . . . 10-16 [1]
10.9.5 Translation Page Index Register . . . . . 10-18 [1]
10.9.6 Translation Fault Page Address Register . . . . . 10-18 [1]
10.9.7 MMU Register Address Ranges . . . . . 10-19 [1]

11 Data Memory Unit (DMU) . . . . . 11-1 [1]
11.1 SRAM Redundancy Control . . . . . 11-2 [1]
11.2 DMU SRAM Redundancy Register Programming . . . . . 11-2 [1]
11.3 CPU and CAN SRAM Configuration Register Programming . . . . . 11-4 [1]
11.3.1 Functional Description . . . . . 11-4 [1]

11.3.2 Reading CSCADOUT . . . . . 11-5 [1]
11.4 Soft-Error Detection . . . . . 11-6 [1]
11.5 DMU Registers . . . . . 11-7 [1]
11.5.1 DMU SRAM Redundancy Registers . . . . . 11-10 [1]
11.5.2 CPU SRAM Configuration Registers . . . . . 11-11 [1]
11.5.3 Soft-Error Detection Register . . . . . 11-13 [1]
11.5.4 DMU Register Address Ranges . . . . . 11-14 [1]

12 Memory Protection System . . . . . 12-1 [1]
12.1 Memory Protection Overview . . . . . 12-1 [1]
12.2 Memory Protection Registers . . . . . 12-3 [1]
12.2.1 PSW Protection Fields . . . . . 12-7 [1]
12.2.2 Data Memory Protection Register . . . . . 12-11 [1]
12.2.3 Code Memory Protection Register . . . . . 12-14 [1]
12.3 Sample Protection Register Set . . . . . 12-17 [1]
12.4 Memory Access Checking . . . . . 12-18 [1]
12.4.1 Permitted versus Valid Accesses . . . . . 12-18 [1]
12.4.2 Crossing Protection Boundaries . . . . . 12-19 [1]
12.5 Memory Protection Register Address Ranges . . . . . 12-20 [1]

13 GPIO Ports and Peripheral I/O . . . . . 13-1 [1]
13.1 General Port Operation . . . . . 13-2 [1]
13.2 Port Kernel Registers . . . . . 13-4 [1]
13.2.1 Port Output Register . . . . . 13-5 [1]
13.2.2 Port Input Register . . . . . 13-6 [1]
13.2.3 Direction Register . . . . . 13-7 [1]
13.2.4 Open Drain Control Register . . . . . 13-8 [1]
13.2.5 Pull-Up/Pull-Down Device Register . . . . . 13-9 [1]
13.2.6 Alternate Input Functions . . . . . 13-11 [1]
13.2.7 Alternate Output Functions . . . . . 13-11 [1]
13.3 Port Implementation . . . . . 13-12 [1]
13.3.1 Port 0 . . . . . 13-12 [1]
13.3.1.1 Overview . . . . . 13-12 [1]
13.3.1.2 Port 0 Functions . . . . . 13-13 [1]
13.3.2 Port 1 . . . . . 13-17 [1]
13.3.2.1 Overview . . . . . 13-17 [1]
13.3.2.2 Port 1 Functions . . . . . 13-18 [1]
13.3.3 Port 2 . . . . . 13-22 [1]
13.3.3.1 Overview . . . . . 13-22 [1]
13.3.3.2 Port 2 Functions . . . . . 13-23 [1]
13.3.4 Port 3 . . . . . 13-27 [1]
13.3.4.1 Overview . . . . . 13-27 [1]
13.3.4.2 Port 3 Functions . . . . . 13-28 [1]
13.3.5 Port 4 . . . . . 13-32 [1]

13.3.5.1 Overview . . . 13-32 [1]
13.3.5.2 Port 4 Functions . . . 13-33 [1]
13.4 Port Register Address Map . . . 13-35 [1]

14 External Bus Unit . . . 14-1 [1]
14.1 Overview . . . 14-2 [1]
14.2 EBU Features . . . 14-3 [1]
14.3 Basic EBU Operation . . . 14-4 [1]
14.4 EBU Signal Description . . . 14-7 [1]
14.4.1 Address Bus, A[23:0] . . . 14-8 [1]
14.4.2 Address/Data Bus, AD[31:0] . . . 14-8 [1]
14.4.3 Read/Write Strobes, $\overline{RD}$ and RD/$\overline{WR}$ . . . 14-8 [1]
14.4.4 Address Latch Enable, $\overline{ALE}$ . . . 14-9 [1]
14.4.5 Byte Control Signals, $\overline{BCx}$ . . . 14-9 [1]
14.4.6 Variable Wait State Control, $\overline{WAIT}$ . . . 14-9 [1]
14.4.7 Chip Select Lines, $\overline{CSx}$, $\overline{CSGLB}$ . . . 14-10 [1]
14.4.8 EBU Arbitration Signals, $\overline{HOLD}$, $\overline{HLDA}$ and $\overline{BREQ}$ . . . 14-11 [1]
14.4.9 Emulation Support Signals, $\overline{CSEMU}$ and $\overline{CSOVL}$ . . . 14-11 [1]
14.5 Arbitration . . . 14-12 [1]
14.5.1 External Bus Modes . . . 14-12 [1]
14.5.1.1 Owner Mode . . . 14-12 [1]
14.5.1.2 Hold Mode . . . 14-12 [1]
14.5.2 Arbitration Signals . . . 14-12 [1]
14.5.2.1 Synchronous Arbitration Input Signal Sampling . . . 14-14 [1]
14.5.2.2 Asynchronous Arbitration Input Signal Sampling . . . 14-14 [1]
14.5.3 Arbitration Modes . . . 14-14 [1]
14.5.3.1 No Bus . . . 14-14 [1]
14.5.3.2 EBU is Sole Master . . . 14-15 [1]
14.5.3.3 EBU is Arbiter . . . 14-15 [1]
14.5.3.4 EBU is Participant . . . 14-19 [1]
14.5.4 Locking the External Bus . . . 14-22 [1]
14.5.5 EBU Reaction to an LMB Access to the External Bus . . . 14-23 [1]
14.5.5.1 Pending Access Time-out . . . 14-24 [1]
14.6 EBU Start Up . . . 14-25 [1]
14.6.1 Disabled . . . 14-25 [1]
14.6.2 Emulation Mode . . . 14-25 [1]
14.6.3 Boot Operation . . . 14-25 [1]
14.6.3.1 Boot Memory Type . . . 14-26 [1]
14.6.3.2 Boot Process . . . 14-26 [1]
14.6.3.3 Boot Configuration Value . . . 14-27 [1]
14.7 Emulation Support . . . 14-29 [1]
14.7.1 Emulation Boot . . . 14-29 [1]
14.7.2 Overlay Memory . . . 14-29 [1]

14.8 EBU Operation . . . . . 14-32 [1]
14.8.1 EBU Address Regions . . . . . 14-32 [1]
14.8.1.1 Address Region Selection . . . . . 14-33 [1]
14.8.1.2 Address Region Parameters . . . . . 14-37 [1]
14.8.2 LMB Bus Width Translation . . . . . 14-38 [1]
14.8.3 External Bus Clock Generation . . . . . 14-39 [1]
14.8.4 Address Alignment During Bus Accesses . . . . . 14-40 [1]
14.8.5 Read/Modify/Write Accesses . . . . . 14-40 [1]
14.8.6 Driver Turn-Around Wait States . . . . . 14-41 [1]
14.8.7 Data Buffering . . . . . 14-42 [1]
14.8.8 Data Width of External Devices . . . . . 14-43 [1]
14.8.9 Basic Access Timing . . . . . 14-44 [1]
14.8.9.1 Standard Access Phases . . . . . 14-44 [1]
14.9 Asynchronous Devices . . . . . 14-50 [1]
14.9.1 Features . . . . . 14-50 [1]
14.9.2 Signal List . . . . . 14-51 [1]
14.9.3 Multiple Non-Multiplexed Device Configurations . . . . . 14-52 [1]
14.9.3.1 16-Bit Non-Multiplexed Device Configuration . . . . . 14-53 [1]
14.9.3.2 32-Bit Non-Multiplexed Device Configuration . . . . . 14-53 [1]
14.9.4 Access to Demultiplexed Devices . . . . . 14-54 [1]
14.9.5 Support for Multiple Multiplexed Device Configurations . . . . . 14-56 [1]
14.9.5.1 16-Bit Multiplexed Memory/Peripheral Configuration . . . . . 14-57 [1]
14.9.5.2 32-Bit Multiplexed Memory/Peripheral Configuration . . . . . 14-57 [1]
14.9.5.3 WinCE 32-Bit Multiplexed Memory/Peripheral Configuration . 14-58 [1]
14.9.5.4 Twin 16-Bit Multiplexed Device Configuration . . . . . 14-59 [1]
14.9.6 Access to Multiplexed Devices . . . . . 14-59 [1]
14.9.7 Interfacing to Asynchronous Devices . . . . . 14-62 [1]
14.9.7.1 External Extension of the Command Phase $\overline{\text{WAIT}}$ . . . . . 14-63 [1]
14.9.8 Interfacing to Intel-Style Devices . . . . . 14-65 [1]
14.10 Burst Flash Devices Access . . . . . 14-67 [1]
14.10.1 Features . . . . . 14-68 [1]
14.10.2 Signal List . . . . . 14-68 [1]
14.10.3 Support for two Burst Flash Device Types . . . . . 14-69 [1]
14.10.4 BFCLKO Output . . . . . 14-69 [1]
14.10.4.1 BFCLKO Ungated Mode . . . . . 14-69 [1]
14.10.4.2 BFCLKO Gated Mode (default) . . . . . 14-69 [1]
14.10.5 Burst Flash Configurations . . . . . 14-70 [1]
14.10.5.1 16-Bit Multiplexed Burst Flash Configuration . . . . . 14-73 [1]
14.10.5.2 32-Bit Multiplexed Burst Flash Configuration . . . . . 14-74 [1]
14.10.5.3 Twin 16-Bit Multiplexed Burst Flash Configuration . . . . . 14-75 [1]
14.10.5.4 16-Bit Non-Multiplexed Burst Flash Configuration . . . . . 14-76 [1]
14.10.5.5 32-Bit Non-Multiplexed Burst Flash Configuration . . . . . 14-77 [1]
14.10.6 Standard Access Phases . . . . . 14-77 [1]

14.10.7 Burst Length Control . . . . . 14-78 [1]
14.10.8 Control of ADV and BAA Delays During Burst Flash Access . . . 14-78 [1]
14.10.9 Burst Flash Clock Feedback . . . . . 14-79 [1]
14.10.10 Cycle Definitions of Burst Mode Timing . . . . . 14-80 [1]
14.10.11 External Cycle Control via the WAIT Input . . . . . 14-82 [1]
14.10.11.1 Wait for Page Load Mode (Intel) . . . . . 14-83 [1]
14.10.11.2 Terminate and Start New Burst Mode (AMD) . . . . . 14-84 [1]
14.10.12 Termination of a Burst Access . . . . . 14-85 [1]
14.10.13 Programmable Parameters . . . . . 14-86 [1]
14.11 SDRAM Interface . . . . . 14-88 [1]
14.11.1 SDRAM Signal List . . . . . 14-89 [1]
14.11.2 External Interface . . . . . 14-90 [1]
14.11.3 Supported SDRAM Commands . . . . . 14-90 [1]
14.11.4 Power-Up Sequence . . . . . 14-92 [1]
14.11.5 Initialization Sequence . . . . . 14-92 [1]
14.11.6 SDRAM Burst Accesses . . . . . 14-95 [1]
14.11.7 Multibanking Operation . . . . . 14-98 [1]
14.11.7.1 Bank-Page Tag Structure . . . . . 14-98 [1]
14.11.7.2 Bank Mask and Page Mask . . . . . 14-98 [1]
14.11.7.3 Decisions over Page-hit and Bank-hit . . . . . 14-100 [1]
14.11.8 Banks Precharge . . . . . 14-101 [1]
14.11.9 Refresh Cycles . . . . . 14-102 [1]
14.11.10 Power-Down Support . . . . . 14-103 [1]
14.11.11 SDRAM Addressing Scheme . . . . . 14-104 [1]
14.11.12 SDRAM Clock Gating . . . . . 14-108 [1]
14.12 EBU Registers . . . . . 14-109 [1]
14.12.1 Clock Control Register . . . . . 14-112 [1]
14.12.2 Address Select Registers . . . . . 14-113 [1]
14.12.3 Bus Configuration Registers . . . . . 14-115 [1]
14.12.4 Emulator Configuration Registers . . . . . 14-121 [1]
14.12.5 EBU Configuration Register . . . . . 14-130 [1]
14.12.6 Burst Flash Control Register . . . . . 14-132 [1]
14.12.7 SDRAM Configuration Registers . . . . . 14-136 [1]
14.12.8 USERCON – EBU Test/Control Configuration Register . . . . . 14-143 [1]
14.13 EBULMB Module Implementation . . . . . 14-144 [1]
14.13.1 Interfaces of the EBULMB Modules . . . . . 14-144 [1]
14.13.2 EBULMB Module Related External Registers . . . . . 14-145 [1]
14.13.2.1 Port Control . . . . . 14-145 [1]
14.13.2.2 CSCOMB (CSovl/CSglb) Control . . . . . 14-154 [1]
14.13.3 EBU Register Address Range . . . . . 14-154 [1]

15 Interrupt System . . . . . 15-1 [1]
15.1 Overview . . . . . 15-1 [1]

Table of Contents

15.2 Service Request Nodes . . . . . 15-3 [1]
15.2.1 Service Request Control Registers . . . . . 15-3 [1]
15.2.1.1 Service Request Flag (SRR) . . . . . 15-5 [1]
15.2.1.2 Request Set and Clear Bits (SETR, CLRR) . . . . . 15-5 [1]
15.2.1.3 Enable Bit (SRE) . . . . . 15-5 [1]
15.2.1.4 Service Request Flag (SRR) . . . . . 15-6 [1]
15.2.1.5 Type-of-Service Control (TOS) . . . . . 15-6 [1]
15.2.1.6 Service Request Priority Number (SRPN) . . . . . 15-7 [1]
15.3 Interrupt Control Units . . . . . 15-8 [1]
15.3.1 ICU Interrupt Control Register (ICR) . . . . . 15-8 [1]
15.3.2 Operation of the Interrupt Control Unit (ICU) . . . . . 15-10 [1]
15.4 Arbitration Process . . . . . 15-11 [1]
15.4.1 Controlling the Number of Arbitration Cycles . . . . . 15-11 [1]
15.4.2 Controlling the Duration of Arbitration Cycles . . . . . 15-12 [1]
15.5 Entering an Interrupt Service Routine . . . . . 15-12 [1]
15.6 Exiting an Interrupt Service Routine . . . . . 15-13 [1]
15.7 Interrupt Vector Table . . . . . 15-15 [1]
15.8 Usage of the TC1130 Interrupt System . . . . . 15-18 [1]
15.8.1 Spanning Interrupt Service Routines Across Vector Entries . . . . 15-18 [1]
15.8.2 Configuring Ordinary Interrupt Service Routines . . . . . 15-19 [1]
15.8.3 Interrupt Priority Groups . . . . . 15-19 [1]
15.8.4 Splitting Interrupt Service Across Different Priority Levels . . . . . 15-20 [1]
15.8.5 Using different Priorities for the same Interrupt Source . . . . . 15-21 [1]
15.8.6 Software Initiated Interrupts . . . . . 15-22 [1]
15.8.7 Interrupt Priority 1 . . . . . 15-22 [1]
15.9 CPU Service Request Nodes . . . . . 15-22 [1]
15.10 Ethernet Interrupts . . . . . 15-23 [1]
15.10.1 Functional Description . . . . . 15-23 [1]
15.10.2 Ethernet Interrupt Register Description . . . . . 15-23 [1]
15.11 FPU Interrupts . . . . . 15-26 [1]
15.12 External Request Unit . . . . . 15-28 [1]
15.12.1 Overview . . . . . 15-28 [1]
15.12.1.1 External Request Select Unit . . . . . 15-30 [1]
15.12.1.2 Event Trigger Logic . . . . . 15-31 [1]
15.12.1.3 The Interrupt Gating Logic (Output Channel) . . . . . 15-33 [1]
15.12.2 External Request Unit Implementation . . . . . 15-35 [1]
15.12.3 External Request Unit Registers . . . . . 15-37 [1]
15.13 Service Request Node Table . . . . . 15-52 [1]

16 Trap System . . . . . 16-1 [1]
16.1 Trap System Overview . . . . . 16-1 [1]
16.2 Trap Types . . . . . 16-2 [1]
16.2.1 Synchronous Traps . . . . . 16-4 [1]

16.2.2 Asynchronous Traps . . . . . 16-5 [1]
16.2.3 Hardware Traps . . . . . 16-5 [1]
16.2.4 Software Traps . . . . . 16-5 [1]
16.2.5 Unrecoverable Traps . . . . . 16-5 [1]
16.2.6 Trap Handling . . . . . 16-5 [1]
16.2.7 Trap Vector Format . . . . . 16-5 [1]
16.2.8 Accessing the Trap Vector Table . . . . . 16-6 [1]
16.2.9 Return PC . . . . . 16-6 [1]
16.2.10 Initial State upon a Trap . . . . . 16-7 [1]
16.3 Trap Descriptions . . . . . 16-8 [1]
16.3.1 MMU Traps . . . . . 16-8 [1]
16.3.2 Internal Protection Traps . . . . . 16-8 [1]
16.3.3 Instruction Errors . . . . . 16-10 [1]
16.3.4 Context Management . . . . . 16-11 [1]
16.3.5 System Bus and Peripheral Errors . . . . . 16-13 [1]
16.3.6 Assertion Traps . . . . . 16-14 [1]
16.3.7 System Call . . . . . 16-14 [1]
16.3.8 Non-Maskable Interrupt (NMI) . . . . . 16-14 [1]
16.4 Trap Priorities . . . . . 16-15 [1]
16.5 Trap Vector Table . . . . . 16-17 [1]
16.5.1 Entering a Trap Service Routine . . . . . 16-18 [1]
16.6 Non-Maskable Interrupt . . . . . 16-19 [1]
16.6.1 NMI Status Register . . . . . 16-19 [1]
16.6.1.1 External NMI Input . . . . . 16-21 [1]
16.6.1.2 Phase-Locked Loop NMI . . . . . 16-21 [1]
16.6.1.3 Watchdog Timer NMI . . . . . 16-21 [1]
16.6.1.4 Parity Error NMI . . . . . 16-21 [1]
16.6.1.5 Deep Sleep Mode NMI . . . . . 16-21 [1]

17 Direct Memory Access Controller (DMA) . . . . . 17-1 [1]
17.1 DMA Controller Description . . . . . 17-2 [1]
17.1.1 Features . . . . . 17-2 [1]
17.1.2 Access Types . . . . . 17-3 [1]
17.1.3 Definition of Terms . . . . . 17-3 [1]
17.1.4 DMA Principle . . . . . 17-4 [1]
17.1.5 DMA Block Diagram . . . . . 17-6 [1]
17.1.6 DMA Operation Functionality . . . . . 17-7 [1]
17.1.6.1 Shadow Registers . . . . . 17-7 [1]
17.1.6.2 DMA Channel Request Control . . . . . 17-10 [1]
17.1.6.3 DMA Channel Operation Mode . . . . . 17-10 [1]
17.1.6.4 Move Count . . . . . 17-16 [1]
17.1.6.5 Request Lost . . . . . 17-16 [1]
17.1.6.6 Circular Buffer . . . . . 17-17 [1]

17.1.6.7 Interrupt Generation . . . . . 17-18 [1]
17.1.6.8 Pattern Detection . . . . . 17-19 [1]
17.1.6.9 Error Conditions . . . . . 17-21 [1]
17.1.6.10 Channel Reset Operation . . . . . 17-22 [1]
17.1.6.11 Programmable Address Modification . . . . . 17-23 [1]
17.1.7 Transaction Control Engine . . . . . 17-25 [1]
17.1.8 Switch . . . . . 17-26 [1]
17.1.9 Request Assignment Unit . . . . . 17-29 [1]
17.1.10 On-Chip Debug System (OCDS) . . . . . 17-30 [1]
17.1.11 Trace Signals . . . . . 17-31 [1]
17.1.12 Access Protection . . . . . 17-32 [1]
17.1.13 General Interrupt Structure . . . . . 17-32 [1]
17.2 DMA Module Kernel Registers . . . . . 17-34 [1]
17.2.1 Overview . . . . . 17-34 [1]
17.2.2 System Registers . . . . . 17-36 [1]
17.2.3 General Control and Status Registers . . . . . 17-41 [1]
17.2.4 Move Engine Registers . . . . . 17-54 [1]
17.2.5 Channel Control, Status and Address Registers . . . . . 17-59 [1]
17.3 DMA Module Implementation . . . . . 17-73 [1]
17.3.1 Interfaces of the DMA Module . . . . . 17-73 [1]
17.3.1.1 DMA Request Assignment Matrix . . . . . 17-74 [1]
17.3.1.2 Access Protection . . . . . 17-77 [1]
17.3.2 DMA Implementation Specific Registers . . . . . 17-82 [1]
17.3.2.1 Clock Control Register . . . . . 17-84 [1]
17.3.2.2 DMA Interrupt Registers . . . . . 17-85 [1]
17.3.2.3 MLI Interrupt Registers . . . . . 17-86 [1]
17.3.2.4 System Interrupt Registers . . . . . 17-87 [1]
17.3.2.5 DMA Bus Time-Out Control Register . . . . . 17-88 [1]
17.3.3 Address Map . . . . . 17-89 [1]
17.4 Memory Checker Module . . . . . 17-90 [1]
17.4.1 Functional Description . . . . . 17-90 [1]
17.4.2 Registers . . . . . 17-91 [1]
17.4.3 Address Map . . . . . 17-95 [1]

18 Bus Systems and Bus Bridges . . . . . 18-1 [1]
18.1 Local Memory Bus Overview (LMB) . . . . . 18-1 [1]
18.2 Local Memory Bus Hub . . . . . 18-4 [1]
18.2.1 LMBH Agent Priorities . . . . . 18-4 [1]
18.2.2 LMBH Default Master . . . . . 18-5 [1]
18.2.3 Reset . . . . . 18-5 [1]
18.2.4 LMB Error Capture . . . . . 18-5 [1]
18.2.5 LBCU Registers . . . . . 18-6 [1]
18.3 LMB-to-FPI (LFI) Bus Bridge . . . . . 18-12 [1]

18.3.1 Functional Description . . . . . 18-13 [1]
18.3.2 LFI Configuration Register . . . . . 18-17 [1]
18.4 Flexible Peripheral Interconnect Bus (FPI Bus) . . . . . 18-18 [1]
18.5 System Bus to DMA Bus Bridge . . . . . 18-20 [1]
18.6 Bus Control Units . . . . . 18-21 [1]
18.6.1 Bus Arbitration . . . . . 18-22 [1]
18.6.1.1 Bus Starvation Prevention . . . . . 18-22 [1]
18.6.1.2 Error Handling . . . . . 18-23 [1]
18.6.2 OCDS Debug . . . . . 18-23 [1]
18.6.3 Tag Assignments . . . . . 18-24 [1]
18.6.4 Arbitration Priorities . . . . . 18-25 [1]
18.6.4.1 LMB Bus . . . . . 18-25 [1]
18.6.4.2 System FPI Bus BCU . . . . . 18-25 [1]
18.6.5 SBCU Registers . . . . . 18-27 [1]
18.6.5.1 SBCU Control Register . . . . . 18-28 [1]
18.6.5.2 SBCU Application Error Registers . . . . . 18-30 [1]
18.6.5.3 SBCU OCDS Debug Registers . . . . . 18-33 [1]
18.6.5.4 SBCU Service Request Control Register . . . . . 18-42 [1]

19 System Timer . . . . . 19-1 [1]
19.1 Overview . . . . . 19-1 [1]
19.2 Kernel Functions . . . . . 19-1 [1]
19.2.1 Resolution and Ranges . . . . . 19-3 [1]
19.2.2 Compare Register Operation . . . . . 19-4 [1]
19.2.3 Compare Match Interrupt Control . . . . . 19-5 [1]
19.3 Kernel Registers . . . . . 19-6 [1]
19.4 External Registers . . . . . 19-15 [1]
19.4.1 Clock Control Register . . . . . 19-15 [1]
19.4.2 Interrupt Register . . . . . 19-18 [1]
19.5 STM Register Address Ranges . . . . . 19-19 [1]

20 Watchdog Timer . . . . . 20-1 [1]
20.1 Watchdog Timer Overview . . . . . 20-1 [1]
20.2 Features of the Watchdog Timer . . . . . 20-2 [1]
20.3 The ENDINIT Function . . . . . 20-3 [1]
20.4 Watchdog Timer Operation . . . . . 20-5 [1]
20.4.1 WDT Register Overview . . . . . 20-6 [1]
20.4.2 Modes of the Watchdog Timer . . . . . 20-7 [1]
20.4.2.1 Time-out Mode . . . . . 20-8 [1]
20.4.2.2 Normal Mode . . . . . 20-8 [1]
20.4.2.3 Disable Mode . . . . . 20-8 [1]
20.4.2.4 Prewarning Mode . . . . . 20-9 [1]
20.4.3 Password Access to WDT_CON0 . . . . . 20-10 [1]
20.4.4 Modify Access to WDT_CON0 . . . . . 20-11 [1]

20.4.5 Term Definitions for WDT_CON0 Accesses . . . . . 20-12 [1]
20.4.6 Detailed Descriptions of the WDT Modes . . . . . 20-13 [1]
20.4.6.1 Time-out Mode Details . . . . . 20-13 [1]
20.4.6.2 Normal Mode Details . . . . . 20-14 [1]
20.4.6.3 Disable Mode Details . . . . . 20-15 [1]
20.4.6.4 Prewarning Mode Details . . . . . 20-16 [1]
20.4.6.5 WDT Operation During Power-Saving Modes . . . . . 20-17 [1]
20.4.6.6 WDT Operation in OCDS Suspend Mode . . . . . 20-17 [1]
20.4.7 Determining WDT Periods . . . . . 20-18 [1]
20.4.7.1 Time-out Period . . . . . 20-18 [1]
20.4.7.2 Normal Period . . . . . 20-20 [1]
20.4.7.3 WDT Period During Power-Saving Modes . . . . . 20-20 [1]
20.5 Handling the Watchdog Timer . . . . . 20-22 [1]
20.5.1 System Initialization . . . . . 20-22 [1]
20.5.2 Re-opening Access to Critical System Registers . . . . . 20-23 [1]
20.5.3 Servicing the Watchdog Timer . . . . . 20-23 [1]
20.5.4 Handling the User-Definable Password Field . . . . . 20-24 [1]
20.5.5 Determining the Required Values for a WDT Access . . . . . 20-27 [1]
20.6 Watchdog Timer Registers . . . . . 20-28 [1]
20.6.1 Watchdog Timer Control Register 0 . . . . . 20-29 [1]
20.6.2 Watchdog Timer Control Register 1 . . . . . 20-31 [1]
20.6.3 Watchdog Timer Status Register . . . . . 20-32 [1]

21 On-Chip Debug Support . . . . . 21-1 [1]
21.1 Overview . . . . . 21-1 [1]
21.2 TriCore Debug Support . . . . . 21-4 [1]
21.2.1 OCDS Level 1 Setup . . . . . 21-4 [1]
21.2.2 Feature List . . . . . 21-5 [1]
21.2.3 Debug Events Generation . . . . . 21-6 [1]
21.2.3.1 Assertion of an External Pin . . . . . 21-6 [1]
21.2.3.2 Execution of a Debug Instruction . . . . . 21-6 [1]
21.2.3.3 Execution of an MTCR/MFCR Instruction . . . . . 21-7 [1]
21.2.3.4 Debug Event Generation Unit . . . . . 21-7 [1]
21.2.3.5 Action on a Debug Event . . . . . 21-10 [1]
21.3 OCDS Level 2 Trace Port . . . . . 21-14 [1]
21.3.1 Overview . . . . . 21-14 [1]
21.3.2 Pipeline Status Signals . . . . . 21-14 [1]
21.3.3 Synchronizing with the Status and Indirect Streams . . . . . 21-16 [1]
21.3.4 Indirect Addresses . . . . . 21-17 [1]
21.3.5 Indirect Sync . . . . . 21-17 [1]
21.3.6 Example . . . . . 21-18 [1]
21.3.7 Breakpoint Qualification . . . . . 21-19 [1]
21.4 OCDS System Control Unit (OSCU) . . . . . 21-21 [1]

21.4.1 OCNTRL, OSTATE, OEC and OJCONF Registers . . . . . 21-21 [1]
21.4.2 Operational Overview . . . . . 21-25 [1]
21.4.2.1 OCDS Enabling . . . . . 21-25 [1]
21.4.2.2 Trace Enable Control . . . . . 21-26 [1]
21.4.2.3 OCDS Reset . . . . . 21-27 [1]
21.4.2.4 CPU Halt after Reset . . . . . 21-28 [1]
21.4.2.5 Watchdog Timer Control . . . . . 21-29 [1]
21.4.2.6 System Security . . . . . 21-29 [1]
21.4.3 Reset Behavior . . . . . 21-30 [1]
21.4.4 Power Saving . . . . . 21-31 [1]
21.4.5 Interrupt Service Request Node . . . . . 21-31 [1]
21.4.5.1 SRC Register . . . . . 21-31 [1]
21.5 Multi Core Break Switch (MCBS) . . . . . 21-33 [1]
21.5.1 Break Bus Switch . . . . . 21-37 [1]
21.5.1.1 MCDBBS Register . . . . . 21-38 [1]
21.5.1.2 MCDBBSS Register . . . . . 21-39 [1]
21.5.2 Suspend Signal Generation . . . . . 21-41 [1]
21.5.2.1 MCDSSG Register . . . . . 21-42 [1]
21.5.2.2 Suspend Target Control . . . . . 21-43 [1]
21.5.2.3 MCDSSGC Register . . . . . 21-44 [1]
21.5.2.4 Break to Suspend Converter . . . . . 21-45 [1]
21.5.3 Application Hints . . . . . 21-46 [1]
21.5.3.1 Concurrent Halt and Resume . . . . . 21-46 [1]
21.5.3.2 Suspend and Restart Rules . . . . . 21-46 [1]
21.5.4 Port Logic of Break Pins . . . . . 21-47 [1]
21.6 JTAG-based Debug Interface (Cerberus JDI) . . . . . 21-48 [1]
21.6.1 Introduction . . . . . 21-48 [1]
21.6.2 Cerberus Registers . . . . . 21-50 [1]
21.6.3 Serial Bit Stream Syntax (TDI, TDO Pins) . . . . . 21-55 [1]
21.6.4 I/O Client Instructions . . . . . 21-56 [1]
21.6.5 Shift Register Behavior . . . . . 21-57 [1]
21.6.6 Data Transfer Examples . . . . . 21-58 [1]
21.6.7 RW Mode . . . . . 21-60 [1]
21.6.7.1 Data Type Support . . . . . 21-60 [1]
21.6.7.2 Bus Master Interface . . . . . 21-60 [1]
21.6.8 Communication Mode . . . . . 21-61 [1]
21.6.9 Triggered Transfers . . . . . 21-63 [1]
21.6.10 Trace with External Bus Address . . . . . 21-64 [1]
21.6.11 Error Handling . . . . . 21-65 [1]
21.7 Debugger Startup . . . . . 21-66 [1]
21.7.1 Hot Attach . . . . . 21-66 [1]
21.7.2 After Power-On Reset . . . . . 21-66 [1]
21.7.3 With Halt After Reset . . . . . 21-66 [1]

21.7.4 Locked Debugger Interface . . . . . 21-67 [1]
21.7.5 Required Initializations . . . . . 21-68 [1]
21.8 Port Control . . . . . 21-69 [1]
21.9 OCDS (Cerberus) Register Address Ranges . . . . . 21-86 [1]

22 Register Overview . . . . . 22-1 [1]
22.1 Segments 0 - 14 . . . . . 22-2 [1]
22.2 Segment 15 . . . . . 22-5 [1]
22.2.1 Registers . . . . . 22-10 [1]

23 Asynchronous/Synchronous Serial Interface (ASC) . . . . . 23-1 [2]
23.1 ASC Kernel Description . . . . . 23-2 [2]
23.1.1 Overview . . . . . 23-3 [2]
23.1.2 General Operation . . . . . 23-4 [2]
23.1.3 Asynchronous Operation . . . . . 23-5 [2]
23.1.3.1 Asynchronous Data Frames . . . . . 23-6 [2]
23.1.3.2 Asynchronous Transmission . . . . . 23-9 [2]
23.1.3.3 Transmit FIFO Operation . . . . . 23-10 [2]
23.1.3.4 Asynchronous Reception . . . . . 23-12 [2]
23.1.3.5 Receive FIFO Operation . . . . . 23-12 [2]
23.1.3.6 FIFO Transparent Mode . . . . . 23-14 [2]
23.1.3.7 IrDA Mode . . . . . 23-15 [2]
23.1.3.8 RXD/TXD Data Path Selection in Asynchronous Modes . . . . 23-17 [2]
23.1.4 Synchronous Operation . . . . . 23-18 [2]
23.1.4.1 Synchronous Transmission . . . . . 23-19 [2]
23.1.4.2 Synchronous Reception . . . . . 23-19 [2]
23.1.4.3 Synchronous Timing . . . . . 23-19 [2]
23.1.5 Baud Rate Generation . . . . . 23-20 [2]
23.1.5.1 Baud Rate in Asynchronous Mode . . . . . 23-21 [2]
23.1.5.2 Baud Rate in Synchronous Mode . . . . . 23-24 [2]
23.1.6 Hardware Error Detection Capabilities . . . . . 23-25 [2]
23.1.7 Interrupts . . . . . 23-27 [2]
23.2 ASC Kernel Registers . . . . . 23-29 [2]
23.3 ASC0/ASC1/ASC2 Module Implementation . . . . . 23-43 [2]
23.3.1 Interfaces of the ASC Modules . . . . . 23-43 [2]
23.3.2 ASC0/ASC1/ASC2 Module Related External Registers . . . . . 23-45 [2]
23.3.2.1 Clock Control Registers . . . . . 23-45 [2]
23.3.2.2 Peripheral Input Select Register . . . . . 23-47 [2]
23.3.2.3 Port Control . . . . . 23-49 [2]
23.3.2.4 Interrupt Registers . . . . . 23-57 [2]
23.3.3 DMA Requests . . . . . 23-60 [2]
23.3.4 ASC0/ASC1/ASC2 Register Address Ranges . . . . . 23-60 [2]

24 Synchronous Serial Interface (SSC) . . . . . 24-1 [2]

Table of Contents

24.1 SSC Kernel Description . . . . . 24-2 [2]
24.1.1 Overview . . . . . 24-3 [2]
24.1.2 General Operation . . . . . 24-4 [2]
24.1.2.1 Operating Mode Selection . . . . . 24-6 [2]
24.1.2.2 Full-Duplex Operation . . . . . 24-7 [2]
24.1.2.3 Half-Duplex Operation . . . . . 24-10 [2]
24.1.2.4 Continuous Transfers . . . . . 24-11 [2]
24.1.2.5 Port Control . . . . . 24-12 [2]
24.1.2.6 Transmit FIFO Operation . . . . . 24-13 [2]
24.1.2.7 Receive FIFO Operation . . . . . 24-15 [2]
24.1.2.8 FIFO Transparent Mode . . . . . 24-17 [2]
24.1.2.9 Baud Rate Generation . . . . . 24-19 [2]
24.1.2.10 Slave Select Input Operation . . . . . 24-21 [2]
24.1.2.11 Slave Select Output Generation Unit . . . . . 24-23 [2]
24.1.2.12 Shift Clock Generation . . . . . 24-25 [2]
24.1.2.13 Error Detection Mechanisms . . . . . 24-26 [2]
24.2 SSC Kernel Registers . . . . . 24-28 [2]
24.3 SSC0/SSC1 Module Implementation . . . . . 24-45 [2]
24.3.1 Interfaces of the SSC Modules . . . . . 24-45 [2]
24.3.2 SSC0/SSC1 Module Related External Registers . . . . . 24-47 [2]
24.3.3 Clock Control . . . . . 24-48 [2]
24.3.3.1 Port Input Select Register . . . . . 24-52 [2]
24.3.3.2 Port Control . . . . . 24-57 [2]
24.3.3.3 Interrupt Registers . . . . . 24-74 [2]
24.3.4 DMA Requests . . . . . 24-75 [2]
24.3.5 SSC0/SSC1 Register Address Ranges . . . . . 24-75 [2]

25 IIC . . . . . 25-1 [2]
25.1 IIC Kernel Description . . . . . 25-2 [2]
25.1.1 Introduction . . . . . 25-2 [2]
25.1.2 Operational Overview . . . . . 25-3 [2]
25.1.3 Functional Overview . . . . . 25-6 [2]
25.1.3.1 Operation in Master Mode . . . . . 25-6 [2]
25.1.3.2 Operation in Multimaster Mode . . . . . 25-6 [2]
25.1.3.3 Operation in Slave Mode . . . . . 25-6 [2]
25.1.4 Baud Rate Selection . . . . . 25-7 [2]
25.1.5 Interrupts . . . . . 25-9 [2]
25.1.6 Synchronization . . . . . 25-10 [2]
25.1.7 Programming . . . . . 25-10 [2]
25.1.7.1 Initialization . . . . . 25-10 [2]
25.1.7.2 Repeated Start Condition . . . . . 25-10 [2]
25.1.7.3 Start Condition . . . . . 25-10 [2]
25.1.7.4 Sending Data Bytes . . . . . 25-10 [2]

25.1.7.5 Stop Condition . . . . . 25-10 [2]
25.1.7.6 Receiving Data Bytes . . . . . 25-11 [2]
25.2 IIC Kernel Registers . . . . . 25-12 [2]
25.3 IIC Module Implementation . . . . . 25-26 [2]
25.3.1 Interfaces of the IIC Module . . . . . 25-26 [2]
25.3.2 IIC Module Related External Registers . . . . . 25-27 [2]
25.3.2.1 Clock Control Register . . . . . 25-28 [2]
25.3.2.2 Port Registers . . . . . 25-29 [2]
25.3.2.3 Service Request Control Registers . . . . . 25-32 [2]
25.3.3 DMA Requests . . . . . 25-33 [2]
25.3.4 IIC Register Address Range . . . . . 25-33 [2]

26 USB . . . . . 26-1 [2]
26.1 USB Kernel Description . . . . . 26-2 [2]
26.1.1 Introduction . . . . . 26-2 [2]
26.1.2 General Operation . . . . . 26-4 [2]
26.1.2.1 Memory . . . . . 26-6 [2]
26.1.2.2 Access Control . . . . . 26-6 [2]
26.1.2.3 Assembly Buffers . . . . . 26-6 [2]
26.1.2.4 MMUs of USB . . . . . 26-8 [2]
26.1.2.5 Transfer Modes . . . . . 26-10 [2]
26.1.2.6 Initialization of USB . . . . . 26-11 [2]
26.1.2.7 USB Device Framework . . . . . 26-13 [2]
26.1.2.8 Control Transfers . . . . . 26-14 [2]
26.1.2.9 USB Endpoint Buffer Organization . . . . . 26-16 [2]
26.1.2.10 USB Random Memory Access . . . . . 26-18 [2]
26.1.2.11 Reset Behavior . . . . . 26-18 [2]
26.1.2.12 Clock Generation . . . . . 26-19 [2]
26.1.2.13 Clock Control and Power Saving . . . . . 26-20 [2]
26.2 USB Kernel Registers . . . . . 26-21 [2]
26.2.1 System Registers . . . . . 26-26 [2]
26.2.2 Device Registers . . . . . 26-27 [2]
26.2.3 Endpoint Registers . . . . . 26-37 [2]
26.2.4 FIFO Registers . . . . . 26-45 [2]
26.2.5 Device Interrupt Registers . . . . . 26-52 [2]
26.2.6 Endpoint Interrupt Registers . . . . . 26-60 [2]
26.2.7 Interrupt Node Pointer . . . . . 26-65 [2]
26.3 USB Module Implementation . . . . . 26-66 [2]
26.3.1 Interfaces of the USB Modules . . . . . 26-66 [2]
26.3.2 USB Module Related External Registers . . . . . 26-67 [2]
26.3.2.1 Clock Control Registers . . . . . 26-68 [2]
26.3.2.2 Peripheral Input Select Register . . . . . 26-69 [2]
26.3.2.3 Port Control . . . . . 26-69 [2]

26.3.2.4 Interrupt Registers . . . . . 26-74 [2]
26.3.3 DMA Requests . . . . . 26-74 [2]
26.3.4 USB Register Address Ranges . . . . . 26-75 [2]

27 Micro Link Serial Bus Interface (MLI) . . . . . 27-1 [2]
27.1 MLI Kernel Description . . . . . 27-2 [2]
27.1.1 MLI Applications . . . . . 27-2 [2]
27.1.2 Overview . . . . . 27-4 [2]
27.1.2.1 Naming Conventions . . . . . 27-5 [2]
27.1.2.2 MLI Communication Principles . . . . . 27-6 [2]
27.1.3 General Description . . . . . 27-7 [2]
27.1.4 Handshake Description . . . . . 27-13 [2]
27.1.5 Startup Procedure . . . . . 27-15 [2]
27.1.6 MLI Kernel and MLI Interface Logical Connection . . . . . 27-17 [2]
27.1.7 MLI Transmitter . . . . . 27-18 [2]
27.1.7.1 MLI Transmitter Reset . . . . . 27-18 [2]
27.1.7.2 MLI Transmitter Operation Modes . . . . . 27-18 [2]
27.1.7.3 Internal Architecture and Interface Signals . . . . . 27-19 [2]
27.1.7.4 Transmission Format . . . . . 27-20 [2]
27.1.7.5 Transmission Modes . . . . . 27-21 [2]
27.1.7.6 Transfer Mode Selection . . . . . 27-28 [2]
27.1.7.7 Parity Generation . . . . . 27-31 [2]
27.1.7.8 Error Detection and Handling . . . . . 27-31 [2]
27.1.7.9 MLI Transmitter Input/Output Control . . . . . 27-32 [2]
27.1.8 MLI Receiver . . . . . 27-35 [2]
27.1.8.1 MLI Receiver Reset . . . . . 27-35 [2]
27.1.8.2 MLI Receiver Operation Modes . . . . . 27-35 [2]
27.1.8.3 Internal Architecture and Interface Signals . . . . . 27-36 [2]
27.1.8.4 MLI Receiver Operation . . . . . 27-37 [2]
27.1.8.5 Access Protection . . . . . 27-42 [2]
27.1.8.6 Error Handling . . . . . 27-43 [2]
27.1.8.7 MLI Receiver Input/Output Control . . . . . 27-44 [2]
27.1.9 Reading Process Summary . . . . . 27-46 [2]
27.1.10 MLI Interrupts . . . . . 27-47 [2]
27.1.11 Clock Domains and Handshake Timing . . . . . 27-49 [2]
27.1.12 Data Flow Description . . . . . 27-54 [2]
27.1.12.1 Copy Base Address . . . . . 27-54 [2]
27.1.12.2 Command Frame . . . . . 27-55 [2]
27.1.12.3 Write Frame . . . . . 27-56 [2]
27.1.12.4 Read Frame . . . . . 27-57 [2]
27.1.12.5 Access to Remote Window . . . . . 27-58 [2]
27.2 MLI Kernel Registers . . . . . 27-59 [2]
27.2.1 MLI Transmitter Registers . . . . . 27-62 [2]

27.2.2 MLI Receiver Registers . . . . . 27-74 [2]
27.2.3 MLI Kernel Common Registers . . . . . 27-80 [2]
27.2.4 MLI Interrupt Registers . . . . . 27-86 [2]
27.2.5 Memory Protection Registers . . . . . 27-95 [2]
27.3 MLI0/MLI1 Module Implementation . . . . . 27-97 [2]
27.3.1 Interfaces of the MLI Modules . . . . . 27-97 [2]
27.3.1.1 Port Connections of MLI0 . . . . . 27-97 [2]
27.3.1.2 Port Connections of MLI1 . . . . . 27-99 [2]
27.3.2 Access Protection . . . . . 27-101 [2]
27.3.3 MLI0/MLI1 Module Related External Registers . . . . . 27-106 [2]
27.3.3.1 DMA Requests . . . . . 27-106 [2]
27.3.3.2 Interrupt Registers . . . . . 27-107 [2]
27.3.3.3 Fractional Divider Registers . . . . . 27-107 [2]
27.3.3.4 Port Control . . . . . 27-108 [2]
27.3.4 MLI0/MLI1 Register Address Ranges . . . . . 27-119 [2]

28 General Purpose Timer Unit (GPTU) . . . . . 28-1 [2]
28.1 GPTU Kernel Description . . . . . 28-2 [2]
28.1.1 Operational Overview . . . . . 28-3 [2]
28.1.2 Functional Overview . . . . . 28-4 [2]
28.1.2.1 Timers T0 and T1 . . . . . 28-4 [2]
28.1.2.2 Input Selection . . . . . 28-5 [2]
28.1.2.3 Reload Selection . . . . . 28-7 [2]
28.1.2.4 Service Requests, Output Signals, and Trigger Signals . . . . . 28-8 [2]
28.1.2.5 Timers T0 and T1 Configuration Limitations . . . . . 28-9 [2]
28.1.2.6 Timer T2 . . . . . 28-10 [2]
28.1.2.7 Quadrature Counting Mode . . . . . 28-17 [2]
28.1.3 Global GPTU Controls . . . . . 28-18 [2]
28.1.3.1 Output Control . . . . . 28-18 [2]
28.1.3.2 Service Request Control . . . . . 28-20 [2]
28.2 GPTU Kernel Registers . . . . . 28-22 [2]
28.2.1 Timer T0/T1 Registers . . . . . 28-24 [2]
28.2.1.1 Timer T0/T1 Input & Reload Source Selection Register . . . . . 28-24 [2]
28.2.1.2 Timer T0/T1 Output, Trigger, and Service Request Selection Register . . . . . 28-27 [2]
28.2.1.3 Timer T0 and T1 Count and Reload Registers . . . . . 28-29 [2]
28.2.2 Timer T2 Registers . . . . . 28-33 [2]
28.2.2.1 Input Control Registers . . . . . 28-33 [2]
28.2.2.2 Mode Control and Status Register . . . . . 28-38 [2]
28.2.2.3 Timer T0/T1/T2 Run Control Register . . . . . 28-41 [2]
28.2.2.4 T2 Reload/Capture Mode Control Register . . . . . 28-43 [2]
28.2.2.5 Timer T2 Count and Reload/Capture Registers . . . . . 28-45 [2]
28.2.3 Global Control Registers . . . . . 28-47 [2]

28.3 GPTU Module Implementation . . . . . 28-52 [2]
28.3.1 Interfaces of the GPTU Modules . . . . . 28-52 [2]
28.3.2 GPTU Module Related External Registers . . . . . 28-53 [2]
28.3.2.1 Clock Control Registers . . . . . 28-54 [2]
28.3.2.2 Port Control . . . . . 28-55 [2]
28.3.2.3 Interrupt Registers . . . . . 28-61 [2]
28.3.3 GPTU Register Address Ranges . . . . . 28-62 [2]

29 Capture/Compare Unit 6 (CCU6) . . . . . 29-1 [2]
29.1 CCU6 Kernel Description . . . . . 29-2 [2]
29.1.1 Overview . . . . . 29-2 [2]
29.1.2 Timer T12 . . . . . 29-4 [2]
29.1.2.1 Overview . . . . . 29-4 [2]
29.1.2.2 Counting Rules . . . . . 29-5 [2]
29.1.2.3 Switching Rules . . . . . 29-7 [2]
29.1.2.4 Duty Cycle of 0% and 100% . . . . . 29-8 [2]
29.1.2.5 External Timer Start . . . . . 29-8 [2]
29.1.2.6 Compare Mode of T12 . . . . . 29-9 [2]
29.1.2.7 Switching Examples in Edge-aligned Mode . . . . . 29-13 [2]
29.1.2.8 Switching Examples in Center-aligned Mode . . . . . 29-14 [2]
29.1.2.9 Dead-time Generation . . . . . 29-15 [2]
29.1.2.10 Capture Mode . . . . . 29-17 [2]
29.1.2.11 Single Shot Mode . . . . . 29-18 [2]
29.1.2.12 Hysteresis-Like Control Mode . . . . . 29-19 [2]
29.1.3 Timer T13 . . . . . 29-20 [2]
29.1.3.1 Overview . . . . . 29-20 [2]
29.1.3.2 Compare Mode . . . . . 29-21 [2]
29.1.3.3 Single Shot Mode . . . . . 29-22 [2]
29.1.3.4 External Timer Start . . . . . 29-22 [2]
29.1.3.5 Synchronization of T13 to T12 . . . . . 29-23 [2]
29.1.4 Modulation Control . . . . . 29-24 [2]
29.1.5 Trap Handling . . . . . 29-26 [2]
29.1.6 Multi-Channel Mode . . . . . 29-27 [2]
29.1.7 Hall Sensor Mode . . . . . 29-30 [2]
29.1.7.1 Introduction . . . . . 29-30 [2]
29.1.7.2 Sampling of the Hall Pattern . . . . . 29-30 [2]
29.1.7.3 Hall Events . . . . . 29-31 [2]
29.1.7.4 Hall Compare Logic . . . . . 29-32 [2]
29.1.7.5 Brushless-DC Control . . . . . 29-33 [2]
29.1.8 Interrupt Generation . . . . . 29-34 [2]
29.1.9 Suspend Mode . . . . . 29-35 [2]
29.2 CCU6 Kernel Registers . . . . . 29-36 [2]
29.2.1 CCU Control Registers . . . . . 29-38 [2]

29.2.2 Timer12 - Related Registers . . . . . 29-52 [2]
29.2.3 Timer13 - Related Registers . . . . . 29-58 [2]
29.2.4 Modulation Control Registers . . . . . 29-62 [2]
29.2.5 Interrupt Control Registers . . . . . 29-78 [2]
29.3 CCU60/CCU61 Module Implementation . . . . . 29-92 [2]
29.3.1 Interfaces of the CCU6 Modules . . . . . 29-92 [2]
29.3.2 CCU60/CCU61 Module Related External Registers . . . . . 29-94 [2]
29.3.2.1 Clock Control . . . . . 29-95 [2]
29.3.2.2 Clock Control Register . . . . . 29-96 [2]
29.3.2.3 Fractional Divider Register . . . . . 29-97 [2]
29.3.3 Port Control . . . . . 29-98 [2]
29.3.3.1 Service Request Registers . . . . . 29-110 [2]
29.3.4 DMA Requests . . . . . 29-111 [2]
29.3.5 CCU60/CCU61 Register Address Ranges . . . . . 29-111 [2]

30 Controller Area Network (MultiCAN) Controller . . . . . 30-1 [2]
30.1 MultiCAN Kernel Description . . . . . 30-2 [2]
30.1.1 Overview . . . . . 30-2 [2]
30.1.2 Module Structure . . . . . 30-5 [2]
30.1.3 CAN Node Control . . . . . 30-7 [2]
30.1.3.1 Bit Timing . . . . . 30-7 [2]
30.1.3.2 CAN Error Handling . . . . . 30-9 [2]
30.1.3.3 CAN Frame Counter . . . . . 30-9 [2]
30.1.3.4 CAN Node Interrupts . . . . . 30-10 [2]
30.1.4 Message Object List Structure . . . . . 30-11 [2]
30.1.4.1 Basics . . . . . 30-11 [2]
30.1.4.2 List of Unallocated Elements . . . . . 30-12 [2]
30.1.4.3 Connection to the CAN Nodes . . . . . 30-12 [2]
30.1.4.4 List Command Panel . . . . . 30-13 [2]
30.1.5 CAN Node Analysis Features . . . . . 30-15 [2]
30.1.5.1 Analyze Mode . . . . . 30-15 [2]
30.1.5.2 Loop-back Mode . . . . . 30-15 [2]
30.1.5.3 Bit Timing Analysis . . . . . 30-17 [2]
30.1.6 Message Acceptance Filtering . . . . . 30-18 [2]
30.1.6.1 Receive Acceptance Filtering . . . . . 30-18 [2]
30.1.6.2 Transmit Acceptance Filtering . . . . . 30-19 [2]
30.1.7 Message Postprocessing Interface . . . . . 30-21 [2]
30.1.7.1 Message Interrupts . . . . . 30-21 [2]
30.1.7.2 Message Pending . . . . . 30-22 [2]
30.1.8 Message Object Data Handling . . . . . 30-24 [2]
30.1.8.1 Frame Reception . . . . . 30-24 [2]
30.1.8.2 Frame Transmission . . . . . 30-26 [2]
30.1.9 Message Object Functionality . . . . . 30-31 [2]

30.1.9.1 Standard Message Object Mode . . . . . 30-31 [2]
30.1.9.2 Single Data Transfer Mode . . . . . 30-31 [2]
30.1.9.3 Single Transmit Trial . . . . . 30-31 [2]
30.1.9.4 Message Object FIFO Structure . . . . . 30-32 [2]
30.1.9.5 Receive FIFO . . . . . 30-35 [2]
30.1.9.6 Transmit FIFO . . . . . 30-36 [2]
30.1.9.7 Gateway Mode . . . . . 30-37 [2]
30.1.9.8 Foreign Remote Requests . . . . . 30-39 [2]
30.2 MultiCAN Kernel Registers . . . . . 30-40 [2]
30.2.1 Global Module Registers . . . . . 30-42 [2]
30.2.2 CAN Node Registers . . . . . 30-51 [2]
30.2.3 Message Object Registers . . . . . 30-68 [2]
30.2.4 Clock Control . . . . . 30-84 [2]
30.2.5 Suspend Mode . . . . . 30-85 [2]
30.2.6 Interrupt Structure . . . . . 30-86 [2]
30.3 MultiCAN Module Implementation . . . . . 30-87 [2]
30.3.1 Interfaces of the CAN Module . . . . . 30-87 [2]
30.3.2 MultiCAN Module Related External Registers . . . . . 30-88 [2]
30.3.3 Module Clock Generation . . . . . 30-89 [2]
30.3.3.1 Clock Control Register . . . . . 30-91 [2]
30.3.3.2 Fractional Divider Register . . . . . 30-92 [2]
30.3.4 Port Control . . . . . 30-93 [2]
30.3.5 Connection of External Signals . . . . . 30-100 [2]
30.3.6 Service Request Control Registers . . . . . 30-101 [2]
30.3.7 DMA Requests . . . . . 30-101 [2]
30.3.8 MultiCAN Module Register Address Map . . . . . 30-102 [2]

31 Ethernet Controller . . . . . 31-1 [2]
31.1 Ethernet Controller Kernel Description . . . . . 31-2 [2]
31.1.1 Introduction . . . . . 31-2 [2]
31.1.2 Networking Ports . . . . . 31-4 [2]
31.1.2.1 Media Independent Interface for Ethernet . . . . . 31-4 [2]
31.1.2.2 Transmit MII Signals . . . . . 31-6 [2]
31.1.2.3 Receive MII Signals . . . . . 31-7 [2]
31.1.2.4 MII Station Management Signals . . . . . 31-8 [2]
31.1.3 Data Management Unit . . . . . 31-8 [2]
31.1.3.1 Receive Descriptor . . . . . 31-11 [2]
31.1.3.2 Data Management Unit Receive . . . . . 31-14 [2]
31.1.3.3 Transmit Descriptor . . . . . 31-16 [2]
31.1.3.4 Data Management Unit Transmit . . . . . 31-19 [2]
31.1.3.5 Byte Swapping . . . . . 31-21 [2]
31.1.3.6 Interrupt Queues . . . . . 31-21 [2]
31.1.4 Buffer Management . . . . . 31-21 [2]

Table of Contents

31.1.4.1 Internal Receive Buffer . . . . . 31-21 [2]
31.1.4.2 Internal Transmit Buffer . . . . . 31-24 [2]
31.1.5 MAC Controller . . . . . 31-25 [2]
31.1.5.1 100/10-Mbit/s Ethernet MAC Layer Overview . . . . . 31-25 [2]
31.1.5.2 Functional Blocks Overview . . . . . 31-27 [2]
31.1.5.3 Media Independent Interface (MII) . . . . . 31-28 [2]
31.1.5.4 The Transmit Block . . . . . 31-29 [2]
31.1.5.5 Receive Block . . . . . 31-32 [2]
31.1.5.6 Flow Control Block . . . . . 31-34 [2]
31.1.5.7 Detailed Operation . . . . . 31-35 [2]
31.1.6 Interrupt Handling . . . . . 31-46 [2]
31.2 Ethernet Controller Kernel Registers . . . . . 31-49 [2]
31.2.1 Flow Control 100/10 Mbit/s Ethernet MAC Registers . . . . . 31-52 [2]
31.2.2 DMUR Registers . . . . . 31-82 [2]
31.2.3 DMUT Registers . . . . . 31-91 [2]
31.2.4 RB Registers . . . . . 31-99 [2]
31.2.5 TB Registers . . . . . 31-104 [2]
31.3 Ethernet Controller Implementation . . . . . 31-109 [2]
31.3.1 Interfaces of the Ethernet Controller Module . . . . . 31-109 [2]
31.3.2 External Ethernet Controller Module Registers . . . . . 31-110 [2]
31.3.2.1 Port Control Registers . . . . . 31-110 [2]
31.3.3 Interrupt Registers . . . . . 31-115 [2]
31.3.4 Ethernet Controller Register Address Range . . . . . 31-117 [2]

Keyword Index . . . . . L-1 [1+2]

Register Index . . . . . L-16 [1+2]

23 Asynchronous/Synchronous Serial Interface (ASC)

This chapter describes the three ASC asynchronous/synchronous serial interfaces ASC0, ASC1 and ASC2 of the TC1130. It contains the following sections:

- Functional description of the ASC Kernel, valid for ASC0, ASC1 and ASC2 (see **Section 23.1**)
- Register descriptions of all ASC Kernel specific registers (see **Section 23.2**)
- TC1130 implementation specific details and registers of the ASC0/ASC1/ASC2 modules (port connections and control, interrupt control, address decoding, clock control, see **Section 23.3**).

*Note: The ASC kernel register names described in **Section 23.2** will be referenced in the TC1130 User's Manual by the module name prefix "ASC0_" for the ASC0 interface, by "ASC1_" for the ASC1 interface and by "ASC2_" for the ASC2 interface.*

23.1 ASC Kernel Description

Figure 23-1 shows all the functional blocks of the ASC interface.

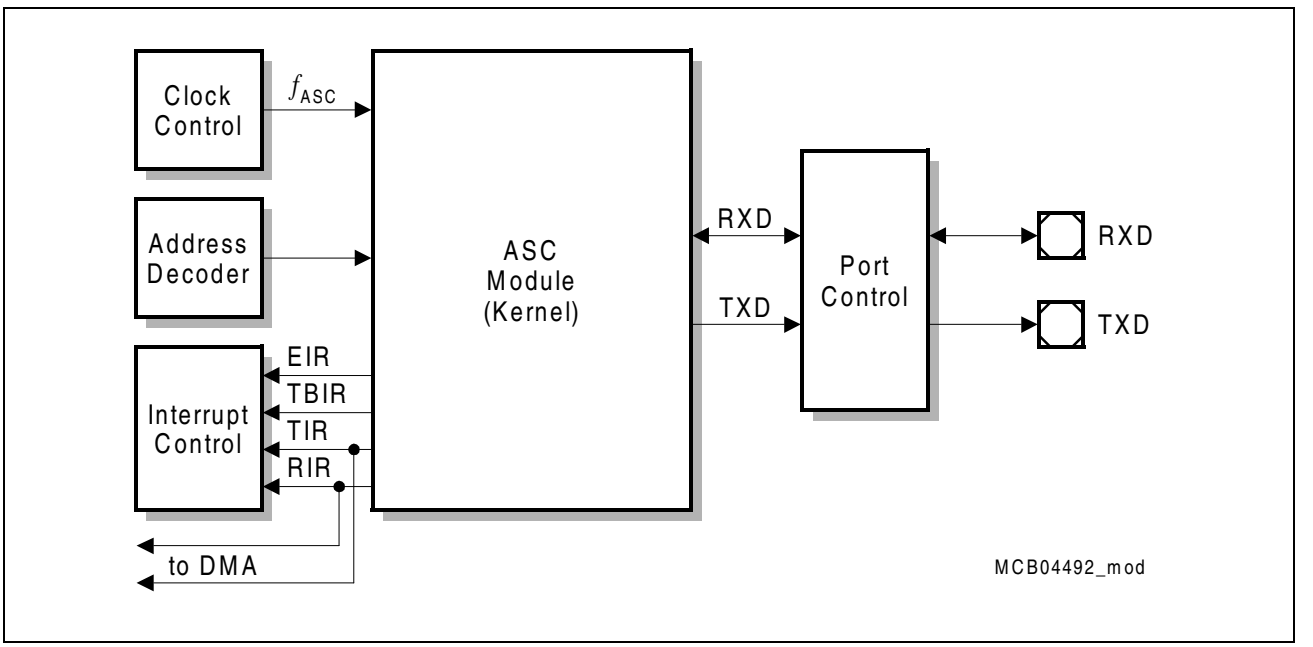


Figure 23-1 General Block Diagram of the ASC Interface

The ASC module communicates with the external world via two I/O lines. The RXD line is the receive data input signal (in synchronous mode also output), and TXD is the transmit output signal.

Clock control, address decoding, and interrupt service request control are managed outside the ASC Module kernel.

23.1.1 Overview

The Asynchronous/Synchronous Serial Interfaces provide serial communication between the TC1130 and other microcontrollers, microprocessors or external peripherals.

The ASC supports full-duplex asynchronous communication and half-duplex synchronous communication. In Synchronous Mode, data is transmitted or received synchronous to a shift clock generated by the ASC internally. In Asynchronous Mode, 8-bit or 9-bit data transfer, parity generation and the number of stop bits can be selected. Parity, framing and overrun error detection are provided to increase the reliability of data transfers. Transmission and reception of data are double-buffered. For multiprocessor communication, a mechanism is included to distinguish address bytes from data bytes. Testing is supported by a loop-back option. A 13-bit baud-rate generator provides the ASC with a separate serial clock signal, which can be accurately adjusted by a prescaler implemented as fractional divider.

Features

- Full-duplex asynchronous operating modes
  - 8-bit or 9-bit data frames, LSB first
  - Parity bit generation/checking
  - One or two stop bits
  - Baud rate from 4.6875 MBaud to 1.12 Baud (@ 75 MHz clock)
- Multiprocessor mode for automatic address/data byte detection
- Loop-back capability
- Half-duplex 8-bit synchronous operating mode
  - Baud rate from 9.375 MBaud to 762.9 Baud (@ 75 MHz clock)
- Support for IrDA data transmission up to 115.2 kBaud maximum
- Double buffered transmitter/receiver
- Interrupt generation
  - On a transmitter buffer empty condition
  - On a transmit last bit of a frame condition
  - On a receiver buffer full condition
  - On an error condition (frame, parity, overrun error)
- FIFO
  - 8-stage receive FIFO (RXFIFO)
  - 8-stage transmit FIFO (TXFIFO)
  - Independent control of RXFIFO and TXFIFO
  - 9-bit FIFO data width
  - Programmable Receive/Transmit Interrupt Trigger Level
  - Receive and Transmit FIFO filling level indication
  - Overrun error generation
  - Underflow error generation

23.1.2 General Operation

The ASC supports full-duplex asynchronous communication up to 4.6875 MBaud and half-duplex synchronous communication up to 9.375 MBaud (@ 75 MHz module clock). In Synchronous Mode, data are transmitted or received synchronous to a shift clock generated by the microcontroller. In Asynchronous Mode, 8-bit or 9-bit data transfer, parity generation, and the number of stop bits can be selected. Parity, framing, and overrun error detection are provided to increase the reliability of data transfers. Transmission and reception of data are double-buffered. For multiprocessor communication, a mechanism is included to distinguish address bytes from data bytes. Testing is supported by a loop-back option. A 13-bit baud rate timer with a versatile input clock divider circuitry provides the ASC with the serial clock signal. In a special asynchronous mode, the ASC supports IrDA data transmission up to 115.2 kBaud with fixed or programmable IrDA pulse width.

A transmission is started by writing to the Transmit Buffer register TBUF. Only the number of data bits determined by the selected operating mode will actually be transmitted, that is, bits written to positions 9 through 15 of register TBUF are always insignificant.

Data transmission is double-buffered, so a new character may be written to the transmit buffer register, before the transmission of the previous character is complete. This allows back-to-back transmission of characters without gaps.

Data reception is enabled by the Receiver Enable Bit CON.REN. After reception of a character has been completed, the received data and, if provided by the selected operating mode, the received parity bit can be read from the (read-only) Receive Buffer register RBUF. Bits in the upper half of RBUF that are not valid in the selected operating mode will be read as zeros.

Data reception is double-buffered, so that reception of a second character may begin before the previously received character has been read out of the receive buffer register. In all modes, receive buffer overrun error detection can be selected through bit CON.OEN. When enabled, the overrun error status flag CON.OE and the error interrupt request line EIR will be activated when the receive buffer register has not been read by the time reception of a second character is complete. The previously received character in the receive buffer is overwritten.

The Loop-Back option (selected by bit CON.LB) allows the data currently being transmitted to be received simultaneously in the receive buffer. This may be used to test serial communication routines at an early stage without having to provide an external network. In Loop-Back Mode, the alternate input/output function of port pins is not required.

23.1.3 Asynchronous Operation

Asynchronous mode supports full-duplex communication, where both transmitter and receiver use the same data frame format and have the same baud rate. Data is transmitted on pin TXD and received on pin RXD. IrDA data transmission/reception is supported up to 115.2 kbit/s. **Figure 23-2** shows the block diagram of the ASC when operating in Asynchronous Mode.

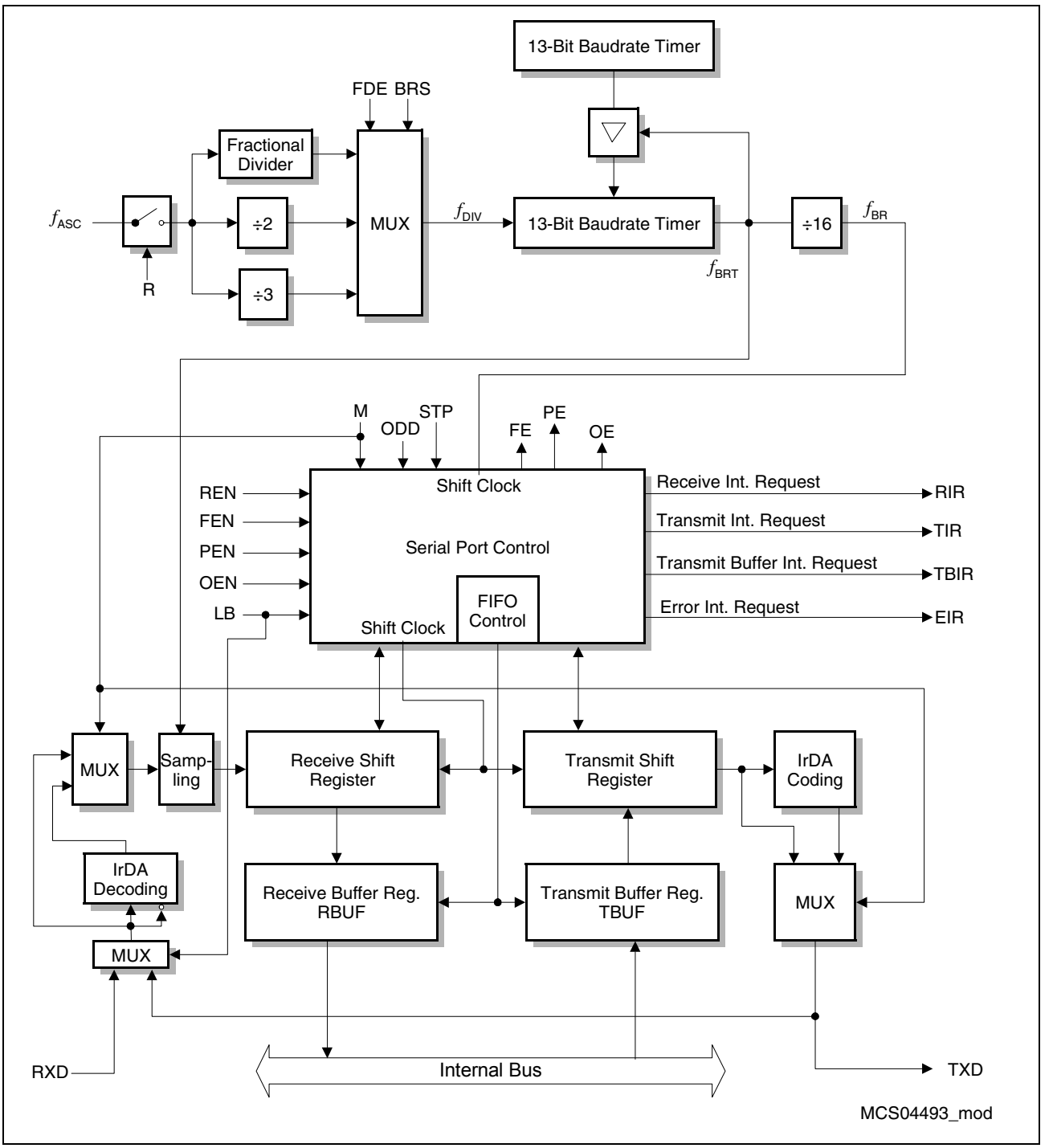


Figure 23-2 Asynchronous Mode of the ASC

23.1.3.1 Asynchronous Data Frames

8-Bit Data Frames

8-bit data frames consist of either eight data bits D7 … D0 (CON.M = 001_B), or of seven data bits D6 … D0 plus an automatically generated parity bit (CON.M = 011_B). Parity may be odd or even, depending on bit CON.ODD. An even parity bit will be set if the modulo-2-sum of the seven data bits is 1. An odd parity bit will be cleared in this case. Parity checking is enabled via bit CON.PEN (always OFF in 8-bit data mode). The parity error flag CON.PE will be set, along with the error interrupt request flag, if a wrong parity bit is received. The parity bit itself will be stored in bit RBUF.7.

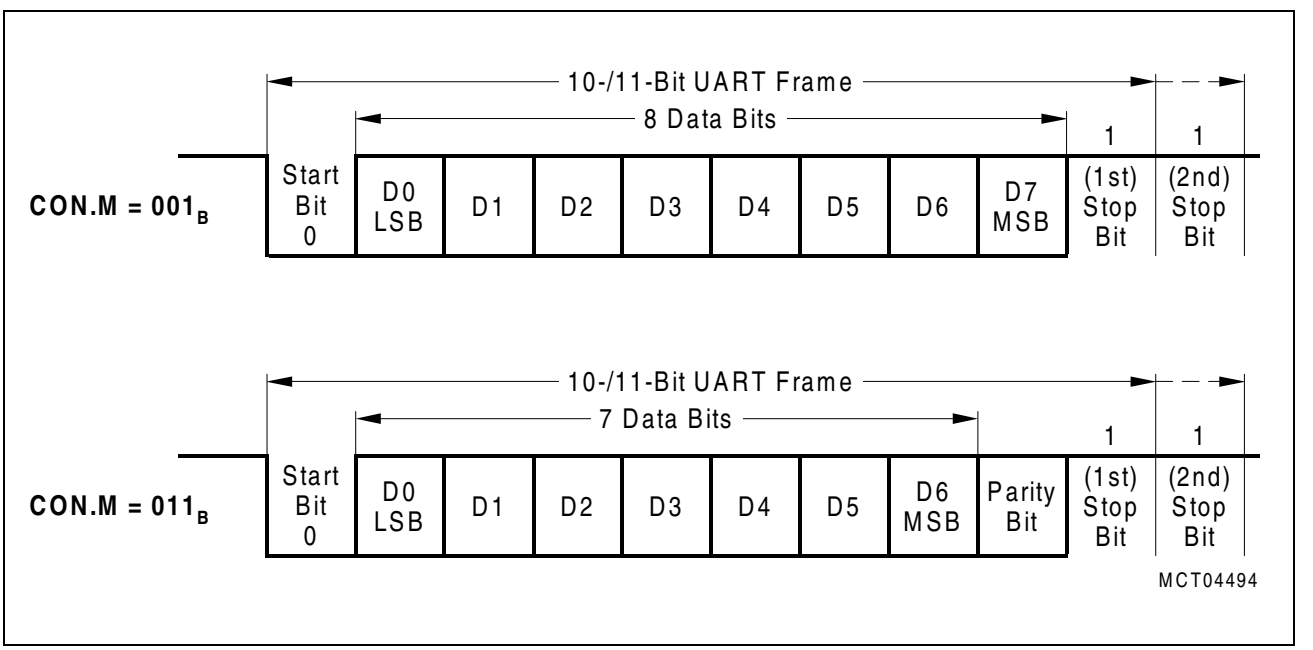


Figure 23-3 Asynchronous 8-Bit Frames

9-Bit Data Frames

9-bit data frames consist of nine data bits D8 … D0 (CON.M = 100_B), or eight data bits D7 … D0 plus an automatically generated parity bit (CON.M = 111_B) or eight data bits D7 … D0 plus wake-up bit (CON.M = 101_B). Parity may be odd or even, depending on bit CON.ODD. An even parity bit will be set if the modulo-2-sum of the eight data bits is 1. An odd parity bit will be cleared in this case. Parity checking is enabled via bit CON.PEN (always OFF in 9-bit data and wake-up mode). The parity error flag CON.PE will be set along with the error interrupt request flag, if a wrong parity bit is received. The parity bit itself will be stored in bit RBUF.8.

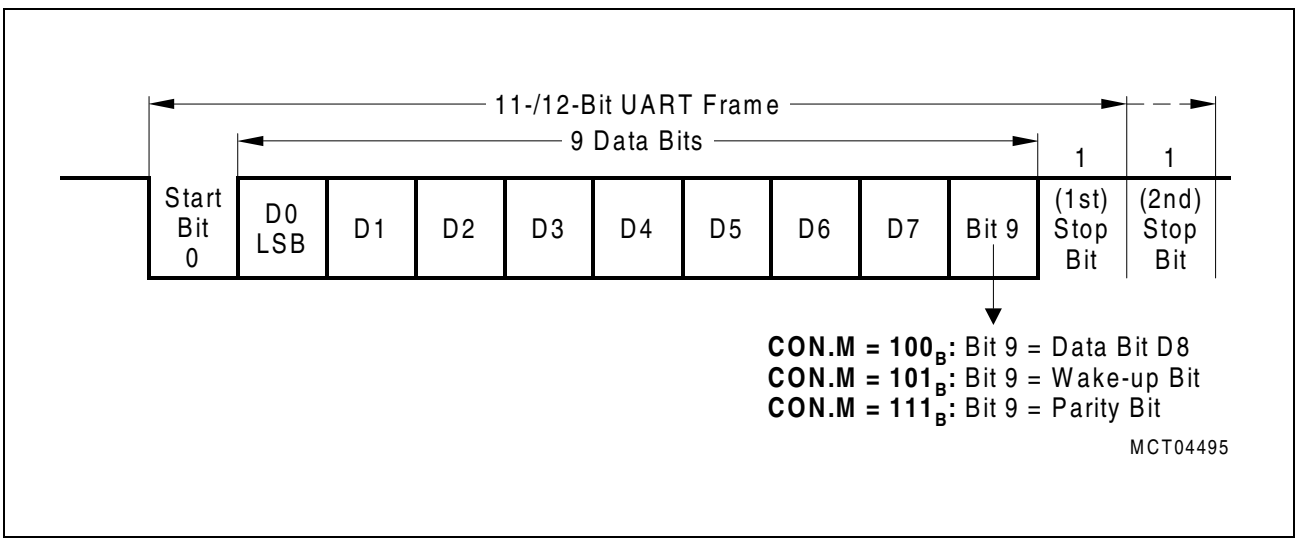


Figure 23-4 Asynchronous 9-Bit Frames

In Wake-up Mode, received frames are transferred to the receive buffer register only if the 9th bit (the wake-up bit) is 1. If this bit is 0, no receive interrupt request will be activated and no data will be transferred.

This feature may be used to control communication in multi-processor systems:
When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte that identifies the target slave. An address byte differs from a data byte in that the additional 9th bit is a 1 for an address byte but is a 0 for a data byte, so, no slave will be interrupted by a data ‘byte’. An address ‘byte’ will interrupt all slaves (operating in 8-bit data + wake-up bit mode), so each slave can examine the eight LSBs of the received character (the address). The addressed slave will switch to 9-bit data mode (for example, by clearing bit CON.M[0]), to enable it to also receive the data bytes that will be coming (having the wake-up bit cleared). The slaves not being addressed remain in 8-bit data + wake-up bit mode, ignoring the following data bytes.

IrDA Frames

The modulation schemes of IrDA are based on standard asynchronous data transmission frames. The asynchronous data format in IrDA mode (CON.M = 010_B) is defined as follows:

- 1 start bit / 8 data bits / 1 stop bit

The coding/decoding of/to the asynchronous data frames is shown in **Figure 23-5**. In general, during IrDA transmissions, UART frames are encoded into IR frames and vice versa. A low-level on the IR frame indicates an “LED off” state. A high level on the IR frame indicates a “LED on” state.

For a 0 bit in the UART frame, a high pulse is generated. For a 1 bit in the UART frame, no pulse is generated. The high pulse starts in the middle of a bit cell and has a fixed width of 3/16 of the bit time. The ASC also allows the length of the IrDA high pulse to be programmed. Further, the polarity of the received IrDA pulse can be inverted in IrDA mode. **Figure 23-5** shows the non-inverted IrDA pulse scheme.

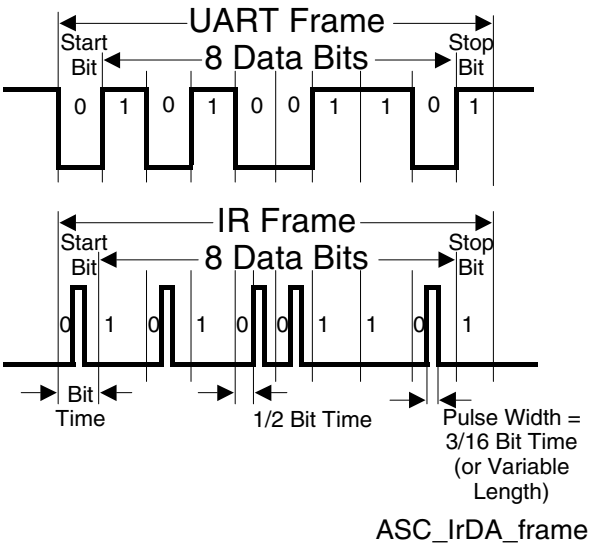


Figure 23-5 IrDA Frame Encoding/Decoding

The ASC IrDA pulse mode/width register PMW contains the 8-bit IrDA pulse width value and the IrDA pulse width mode select bit. This register is required in the IrDA operating mode only.

23.1.3.2 Asynchronous Transmission

Asynchronous transmission begins at the next overflow of the divide-by-16 baud rate timer (transition of the baud rate clock f_{BR}), if bit CON.R must be set and data has been loaded into TBUF. The transmitted data frame consists of three basic elements:

- Start bit
- Data field (eight or nine bits, LSB first, including a parity bit, if selected)
- Delimiter (one or two stop bits)

Data transmission is double-buffered. When the transmitter is idle, the transmit data loaded into the transmit buffer register is immediately moved to the transmit shift register, thus freeing the transmit buffer for the next data to be sent. This is indicated by the transmit buffer interrupt request line TBIR being activated. TBUF may now be loaded with the next data, while transmission of the previous data continues.

The transmit interrupt request line TIR will be activated before the last bit of a frame is transmitted, that is, before the first or the second stop bit is shifted out of the transmit shift register.

Note: The transmitter output pin TXD must be configured for alternate data output.

23.1.3.3 Transmit FIFO Operation

The transmit FIFO (TXFIFO) provides the following functionality:

- Enable/disable control
- Programmable filling level for transmit interrupt generation
- Filling level indication
- FIFO clear (flush) operation
- FIFO overflow error generation

The 8-stage transmit FIFO is controlled by the TXFCON control register. When bit TXFCON.TXFEN is set, the transmit FIFO is enabled. The interrupt trigger level defined by TXFCON.TXFITL defines the filling level of the TXFIFO at which a transmit buffer interrupt TBIR or a transmit interrupt TIR is generated. These interrupts are always generated when the filling level of the transmit FIFO is equal to or less than the value stored in TXFCON.TXFITL.

Bit field FSTAT.TXFFL in the FIFO status register FSTAT indicates the number of entries that are actually written (valid) in the TXFIFO. Therefore, the software can check, in the interrupt service routine, for instance, how many bytes can be still written into the transmit FIFO via register TBUF without getting an overrun error.

The transmit FIFO cannot be accessed directly. All data write operations into the TXFIFO are executed by writing into the TBUF register.

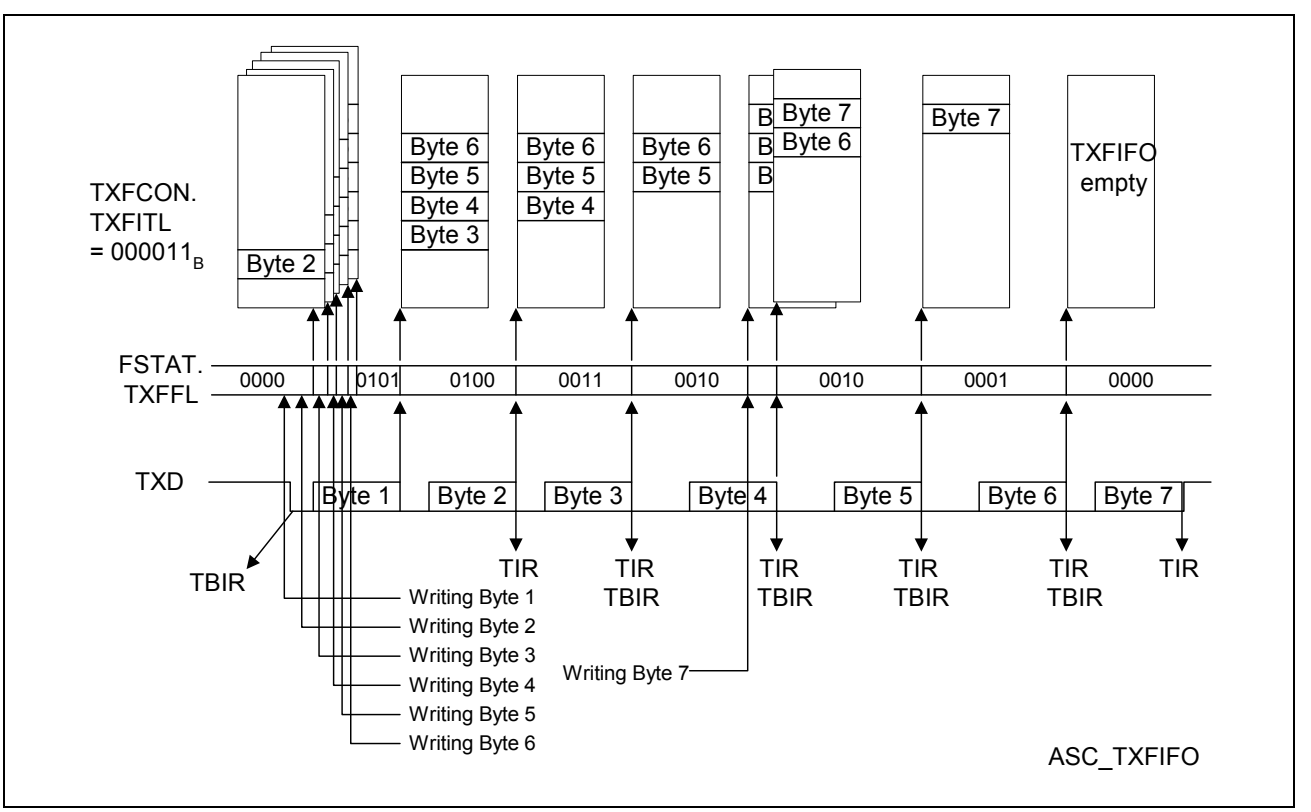


Figure 23-6 Transmit FIFO Operation Example

The example in **Figure 23-6** shows a typical 8-stage transmit FIFO operation. In this example, seven bytes are transmitted via the TXD output line. The transmit FIFO interrupt trigger level TXFCON.TXFITL is set to 000011_B. The first byte written into the empty TXFIFO via TBUF is directly transferred into the transmit shift register and is not written into the FIFO. A transmit buffer interrupt will be generated in this case. After byte 1, bytes 2 to 6 are written into the transmit FIFO.

After the transfer of byte 3 from the TXFIFO into the transmit shift register of the ASC, 3 bytes remain in the TXFIFO. Therefore, the value of TXFCON.TXFITL is reached and a transmit buffer interrupt will be generated at the beginning and a transmit interrupt at the end of the byte 3 serial transmission. During the serial transmission of byte 4, another byte (byte 7) is written into the TXFIFO (TBUF write operation). Finally, after the start of the serial transmission of byte 7, the TXFIFO is again empty.

If the TXFIFO is full and additional bytes are written into TBUF, the error interrupt will be generated with bit CON.OE set. In this case, the data byte that was last written into the transmit FIFO is overwritten and the transmit FIFO filling level FSTAT.TXFFL is set to maximum.

The TXFIFO can be flushed or cleared by setting bit TXFCON.TXFFLU in register TXFCON. After this TXFIFO flush operation, the TXFIFO is empty and the transmit FIFO filling level FSTAT.TXFFL is set to 000000_B. A running serial transmission is not aborted by a receive FIFO flush operation.

Note: The TXFIFO is flushed automatically with a reset operation of the ASC module and if the TXFIFO becomes disabled (resetting bit TXFCON.TXFEN) after it was previously enabled.

23.1.3.4 Asynchronous Reception

Asynchronous reception is initiated by a falling edge (1-to-0 transition) on pin RXD, provided that bits CON.R and CON.REN are set. The receive data input pin RXD is sampled at 16 times the rate of the selected baud rate. A majority decision of the 7th, 8th and 9th sample determines the effective bit value. This avoids erroneous results that may be caused by noise.

If the detected value is not a 0 when the start bit is sampled, the receive circuit is reset and waits for the next 1-to-0 transition at pin RXD. If the start bit proves valid, the receive circuit continues sampling and shifts the incoming data frame into the receive shift register.

When the last stop bit has been received, the contents of the receive shift register are transferred to the receive data buffer register RBUF. Simultaneously, the receive interrupt request line RIR is activated after the 9th sample in the last stop bit timeslot (as programmed), regardless whether valid stop bits have been received or not. The receive circuit then waits for the next start bit (1-to-0 transition) at the receive data input line.

Note: The receive input pin RXD must be configured as input mode.

Asynchronous reception is stopped by clearing bit CON.REN. A currently received frame is completed including generation of the receive interrupt request and an error interrupt request, if appropriate. Start bits that follow this frame will not be recognized.

Note: In wake-up mode received frames are transferred to the receive buffer register only if the 9th bit (the wake-up bit) is 1. If this bit is 0, no receive interrupt request will be activated and no data will be transferred.

23.1.3.5 Receive FIFO Operation

The receive FIFO (RXFIFO) provides the following functionality:

- Enable/disable control
- Programmable filling level for receive interrupt generation
- Filling level indication
- FIFO clear (flush) operation
- FIFO overflow error generation

The 8-stage receive FIFO is controlled by the RXFCON control register. When bit RXFCON.RXFEN is set, the receive FIFO is enabled. The interrupt trigger level defined by RXFCON.RXFITL defines the filling level of RXFIFO at which a receive interrupt RIR is generated. RIR is always generated when the filling level of the receive FIFO is equal to or greater than the value stored in RXFCON.RXFITL.

Bit field FSTAT.RXFFL in the FIFO status register FSTAT indicates the number of bytes that have been actually written into the FIFO and can be read out of the FIFO by a user program.

The receive FIFO cannot be accessed directly. All data read operations from the RXFIFO are executed by reading the RBUF register.

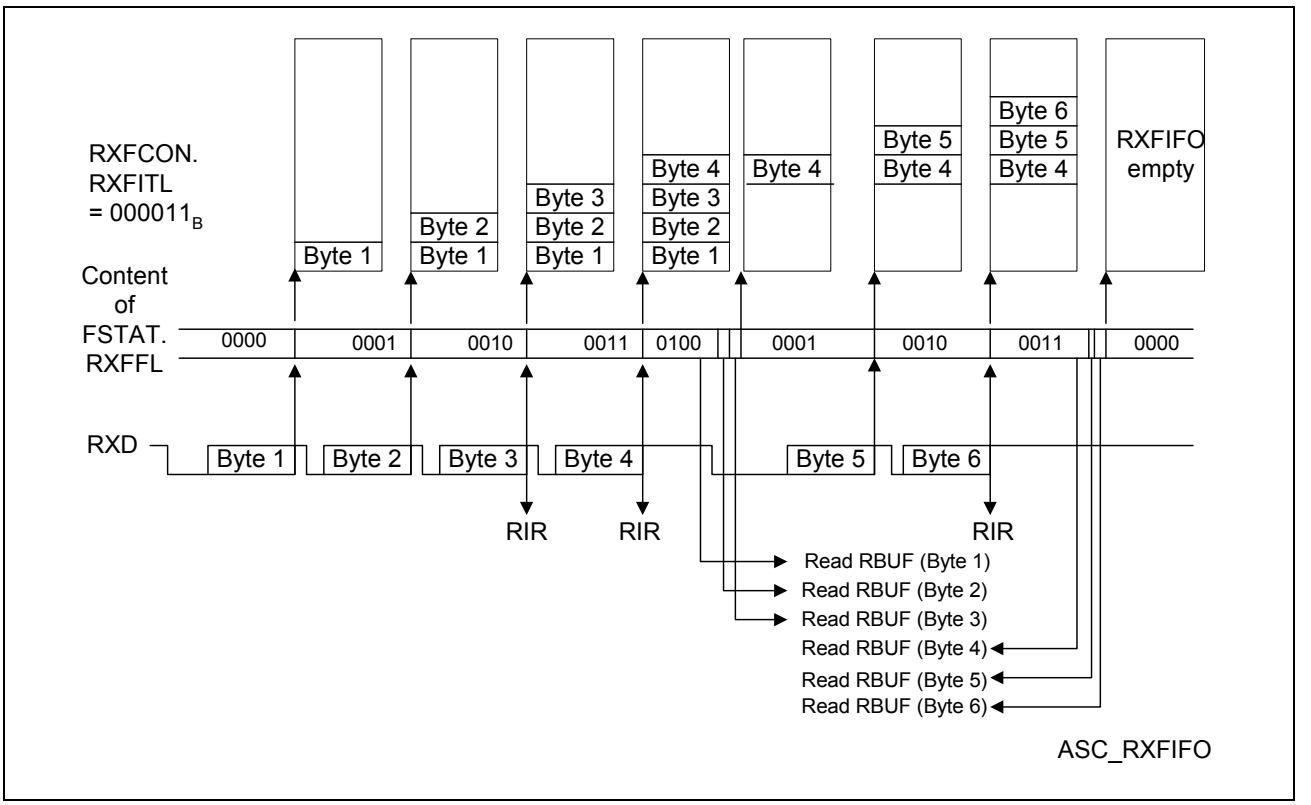


Figure 23-7 Receive FIFO Operation Example

The example in **Figure 23-7** shows a typical 8-stage receive FIFO operation. In this example, six bytes are received via the RXD input line. The receive FIFO interrupt trigger level RXFCON.RXFITL is set to 000011_B. Therefore, the first receive interrupt RIR is generated after the reception of byte 3 (RXFIFO is filled with three bytes).

After the reception of byte 4, three bytes are read out of the receive FIFO. After this read operation, the RXFIFO still contains one byte. RIR becomes again active after two more bytes (byte 5 and 6) have been received (RXFIFO filled again with 3 bytes). Finally, the FIFO is cleared after three read operation.

If the RXFIFO is full and additional bytes are received, the receive interrupt RIR and the error interrupt EIR will be generated with bit CON.OE set. In this case, the data byte last written into the receive FIFO is overwritten. With the overrun condition, the receive FIFO filling level FSTAT.RXFFL is set to maximum. If a RBUF read operation is executed with the RXFIFO enabled but empty, an error interrupt EIR will be generated as well with bit CON.OE set. In this case, the receive FIFO filling level FSTAT.RXFFL is set to 000000_B.

If the RXFIFO is available but disabled (RXFCON.RXFEN = 0) and the receive operation is enabled (CON.REN = 1), the asynchronous receive operation is functionally equivalent to the asynchronous receive operation of the ASC module.

The RXFIFO can be flushed or cleared by setting bit RXFCON.RXFFLU in register RXFCON. After this RXFIFO flush operation, the RXFIFO is empty and the receive FIFO filling level FSTAT.RXFFL is set to 000000_B.

The RXFIFO is flushed automatically with a reset operation of the ASC module and if the RXFIFO becomes disabled (resetting bit RXFCON.RXFEN) after it was previously enabled. Resetting bit CON.REN without resetting RXFCON.RXFEN does not affect (reset) the RXFIFO state. This means that the receive operation of the ASC is stopped, in this case, without changing the content of the RXFIFO. After setting CON.REN again, the RXFIFO with its content is again available.

23.1.3.6 FIFO Transparent Mode

In Transparent Mode, a specific interrupt generation mechanism is used for receive and transmit buffer interrupts. In general, in Transparent Mode, receive interrupts are always generated if data bytes are available in the RXFIFO. Transmit buffer interrupts are always generated if the TXFIFO is not full. The relevant conditions for interrupt generation in Transparent Mode are:

- FIFO filling levels
- Read/write operations on the RBUF/TBUF data register

Interrupt generation for the receive FIFO depends on the RXFIFO filling level and the execution of read operations of register RBUF (see **Figure 23-8**). Transparent Mode for the RXFIFO is enabled when bits RXFCON.RXTMEN and RXFCON.RXFEN in register RXFCON are set.

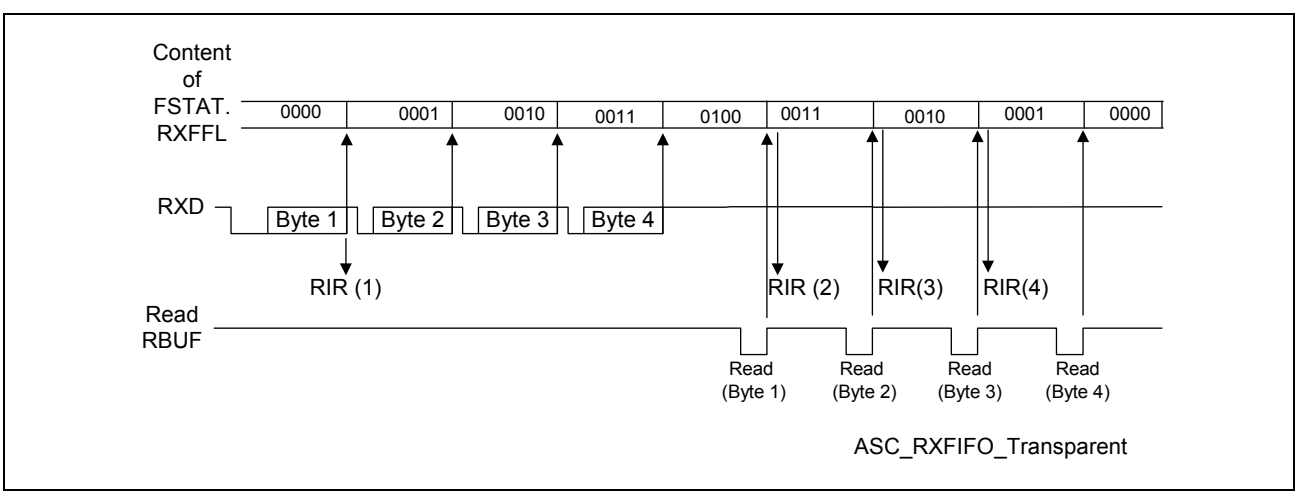


Figure 23-8 Transparent Mode Receive FIFO Operation

If the RXFIFO is empty, a receive interrupt RIR is always generated when the first byte is written into an empty RXFIFO (FSTAT.RXFFL changes from 000000_B to 000001_B). If the RXFIFO is filled with at least one byte, the occurrence of further receive interrupts depends on the read operations of register RBUF. The receive interrupt RIR will always be activated after a RBUF read operation if the RXFIFO still contains data

(FSTAT.RXFFL is not equal to 0000_B). If the RXFIFO is empty after a RBUF read operation, no further receive interrupt will be generated.

If the RXFIFO is full (FSTAT.RXFFL = 1000_B) and additional bytes are received, an error interrupt EIR will be generated with bit CON.OE set. In this case, the data byte last written into the receive FIFO is overwritten. If a RBUF read operation is executed with the RXFIFO enabled but empty (underflow condition), an error interrupt EIR will be generated as well, with bit CON.OE set.

If the RXFIFO is flushed in Transparent Mode, the software must take care that a previous pending receive interrupt is ignored.

Note: The Receive FIFO Interrupt Trigger Level bit field RXFCON.RXFITL is not applicable in Transparent Mode.

Interrupt generation for the transmit FIFO depends on the TXFIFO filling level and the execution of write operations to the register TBUF. Transparent Mode for the TXFIFO is enabled when bits TXFCON.TXTMEN and TXFCON.TXFEN are set.

A transmit buffer interrupt TBIR is always generated when the TXFIFO is not full (FSTAT.TXFFL not equal to 1000_B) after a byte has been written into register TBUF. TBIR is also activated after a TXFIFO flush operation or when the TXFIFO becomes enabled (TXFCON.TXTMEN and TXFCON.TXFEN set) when it was previously disabled. In these cases, the TXFIFO is empty and ready to be filled with data.

If the TXFIFO is full (FSTAT.TXFFL = maximum) and an additional byte is written into TBUF, no further transmit buffer interrupt will be generated after the TBUF write operation. In this case the data byte last written into the transmit FIFO is overwritten and an overrun error interrupt (EIR) will be generated with bit CON.OE set.

Note: The Transmit FIFO Interrupt Trigger Level bit field TXFCON.TXFITL is not applicable in Transparent Mode.

23.1.3.7 IrDA Mode

The duration of the IrDA pulse is normally 3/16 of a bit period. The IrDA standard also allows the pulse duration to be independent of the baud rate or bit period. In this case, the transmitted pulse always has the width corresponding to the 3/16 pulse width at 115.2 kBaud, which is 1.627 µs. Either bit period depended or fixed IrDA pulse width generation can be selected. The IrDA pulse width mode is selected by bit PMW.IRPW.

In case of fixed IrDA pulse width generation, the lower eight bits in register PMW are used to adapt the IrDA pulse width to a fixed value such as 1.627 µs. The fixed IrDA pulse width is generated by a programmable timer as shown in **Figure 23-9**.

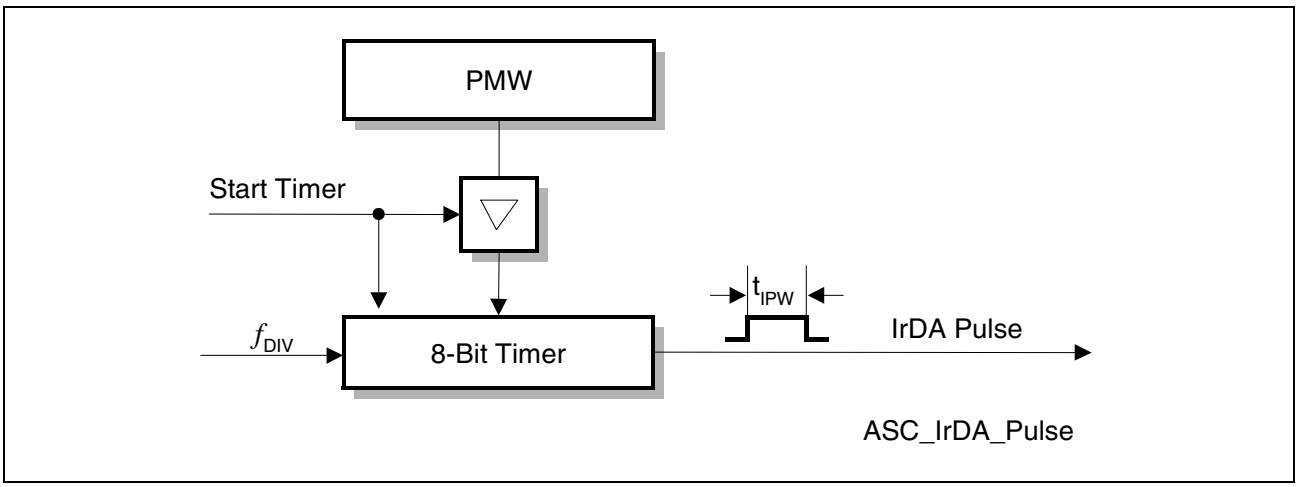


Figure 23-9 Fixed IrDA Pulse Generation

The IrDA pulse width can be calculated according to the formulas given in **Table 23-1**.

Table 23-1 Formulas for the IrDA Pulse Width Calculation

PMW	PMW.IRPW	Formulas	
1 … 255	0	$t_{IPW} = \frac{3}{16 \times Baudrate}$	$t_{IPW\,min} = \frac{(PMW >> 1)}{f_{ASC}}$
	1	$t_{IPW} = \frac{PMW}{f_{ASC}}$	

*Note: The name PMW in the formulas of **Table 23-1** represents the content of the reload register PMW (PW_VALUE), taken as unsigned 8-bit integer.*

The contents of PMW.PW_VALUE further define the minimum IrDA pulse width (t_{IPW} min.) that is still recognized as a valid IrDA pulse during a receive operation. This function is independent of the selected IrDA pulse width mode (fixed or variable), which is defined by bit PMW.IRPW. The minimum IrDA pulse width is calculated by a shift right operation of PMW bit 7-0 by one bit divided by the module clock f_{ASC}.

Note: If PMW.IRPW = 0 (fixed IrDA pulse width), PMW.PW_VALUE must be a value which assures that $t_{IPW} > t_{IPW\,min}$.

Note: PMW value must be carefully selected to guarantee the IrDA pulse is valid. Especially, in some transmission baud rates, the pulse width cannot be too small to be recognized.

Table 23-2 gives some examples for typical frequencies of f_{ASC}.

Table 23-2 IrDA Pulse Width Adaptation to 1.627 µs @115.2 kBaud Rate

f_{ASC}	PMW	t_{IPW}	Error	$t_{IPW\ min}$
25 MHz	41_D	1.64 µs	+0.8%	0.82 µs
40 MHz	65_D	1.625 µs	-0.12%	0.81 µs
48 MHz	78_D	1.625 µs	-0.12%	0.81 µs
50 MHz	81_D	1.62 µs	-0.4%	0.81 µs
75 MHz	122_D	1.6267 µs	-0.02%	0.813 µs

23.1.3.8 RXD/TXD Data Path Selection in Asynchronous Modes

The data paths for the serial input and output data of the Asynchronous mode are affected by control bits in the register CON as shown in **Figure 23-10**. The synchronous mode operation is not affected by these data path selection capabilities.

Two multiplexers are in the RXD input signal path for providing the loopback mode capability (controlled by bit CON.LB) and the IrDA receive pulse inversion capability (controlled by bit CON.RXDI). Depending on the asynchronous operating mode (controlled by bit field CON.M), the ASC output signal or the IrDA coded ASC output signal is switched to the TXD output via a multiplexer.

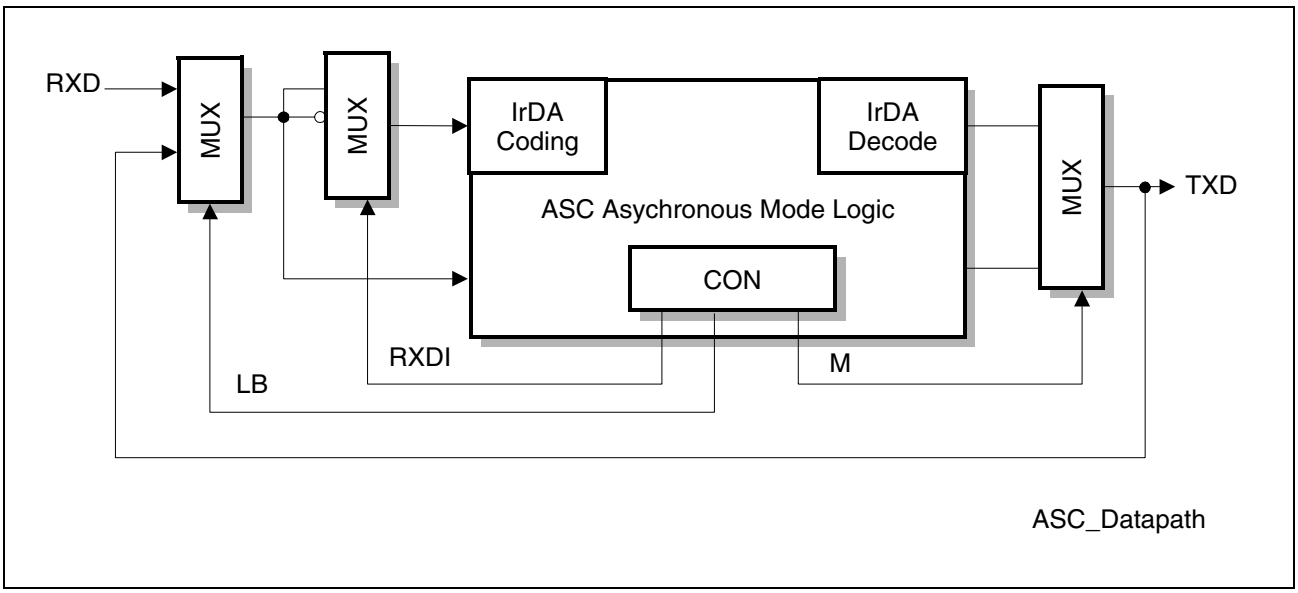


Figure 23-10 RXD/TXD Data Path in Asynchronous Modes

23.1.4 Synchronous Operation

Synchronous Mode supports half-duplex communication, basically for simple I/O expansion via shift registers. Data is transmitted and received via pin RXD while line TXD outputs the shift clock. Synchronous mode is selected with CON.M = 000_B.

Eight data bits are transmitted or received synchronous to a shift clock generated by the internal baud-rate generator. The shift clock is active only as long as data bits are transmitted or received.

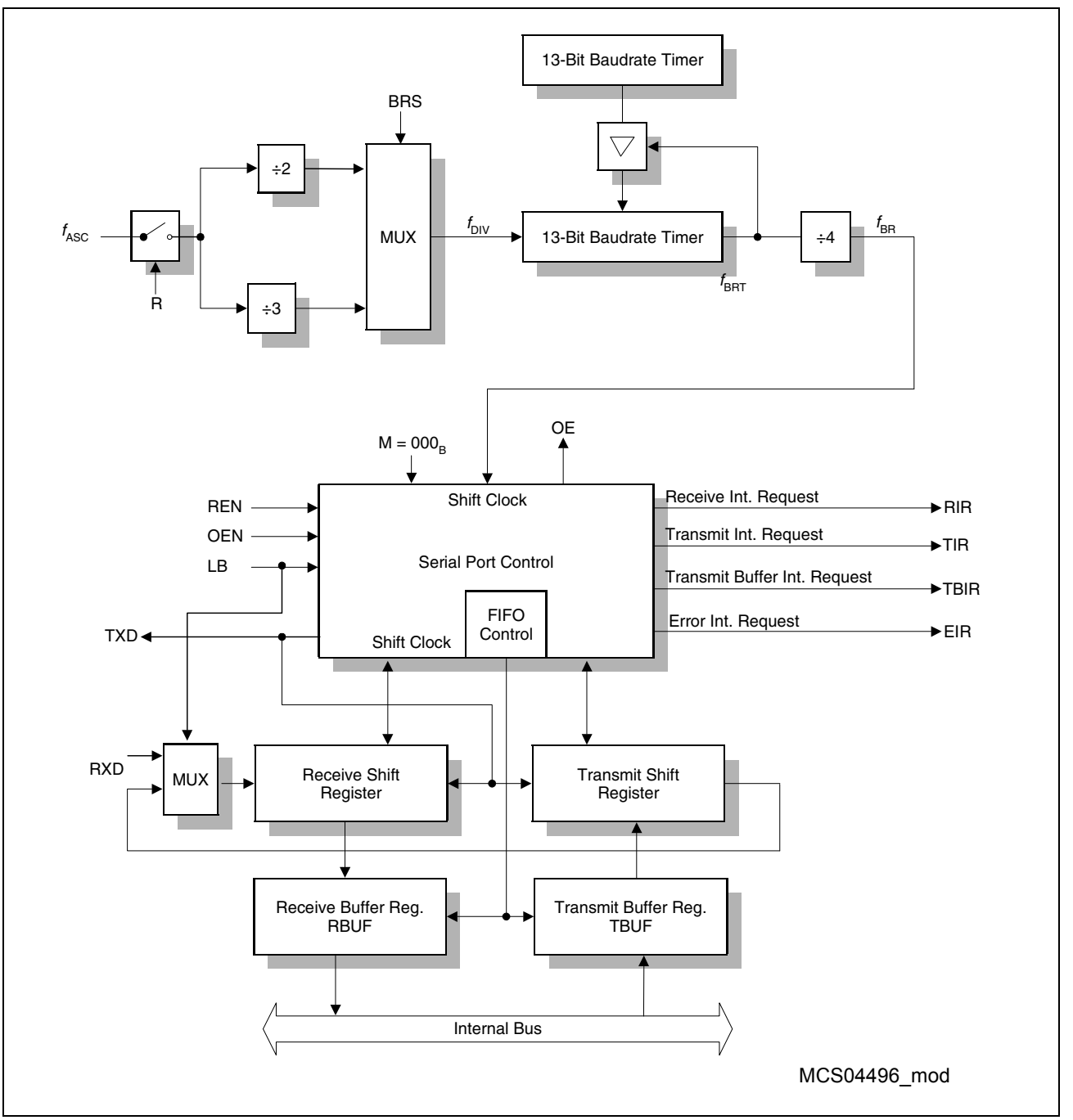


Figure 23-11 Synchronous Mode of Serial Channel ASC

23.1.4.1 Synchronous Transmission

Synchronous transmission begins within four state times after data has been loaded into TBUF, provided that CON.R is set and CON.REN = 0 (half-duplex, no reception). Exception: in Loop-back Mode (bit CON.LB set), CON.REN must be set for reception of the transmitted byte. Data transmission is double buffered. When the transmitter is idle, the transmit data loaded into TBUF is immediately moved to the transmit shift register thus freeing TBUF for more data. This is indicated by the transmit buffer interrupt request line TBIR being activated. TBUF may now be loaded with the next data, while transmission of the previous one continues. The data bits are transmitted synchronous with the shift clock. After the bit time for the 8th data bit, both TXD and RXD will go high, the transmit interrupt request line TIR is activated, and serial data transmission stops.

Pin TXD must be configured for alternate data output in order to provide the shift clock. Pin RXD must also be configured for output during transmission.

23.1.4.2 Synchronous Reception

Synchronous reception is initiated by setting bit CON.REN = 1. If bit CON.R = 1, the data applied at RXD is clocked into the receive shift register synchronous to the clock that is output at pin TXD. After the 8th bit has been shifted in, the contents of the receive shift register are transferred to the receive data buffer RBUF, the receive interrupt request line RIR is activated, the receiver enable bit CON.REN is reset, and serial data reception stops.

Pin TXD must be configured for alternate data output in order to provide the shift clock. Pin RXD must be configured as alternate data input.

Synchronous reception is stopped by clearing bit CON.REN. A currently received byte is completed, including the generation of the receive interrupt request and an error interrupt request, if appropriate. Writing to the transmit buffer register while a reception is in progress has no effect on reception and will not start a transmission.

If a previously received byte has not been read out of the receive buffer register by the time the reception of the next byte is complete, both the error interrupt request line EIR and the overrun error status flag CON.OE will be activated/set, provided that the overrun check has been enabled by bit CON.OEN.

23.1.4.3 Synchronous Timing

Figure 23-12 shows timing diagrams of the ASC Synchronous Mode data reception and data transmission. In Idle State the shift clock is at high level. With the beginning of a synchronous transmission of a data byte, the data is shifted out at RXD with the falling edge of the shift clock. If a data byte is received through RXD, data is latched with the rising edge of the shift clock.

One shift clock cycle (f_{BR}) delay is inserted between two consecutive receive or transmit data bytes.

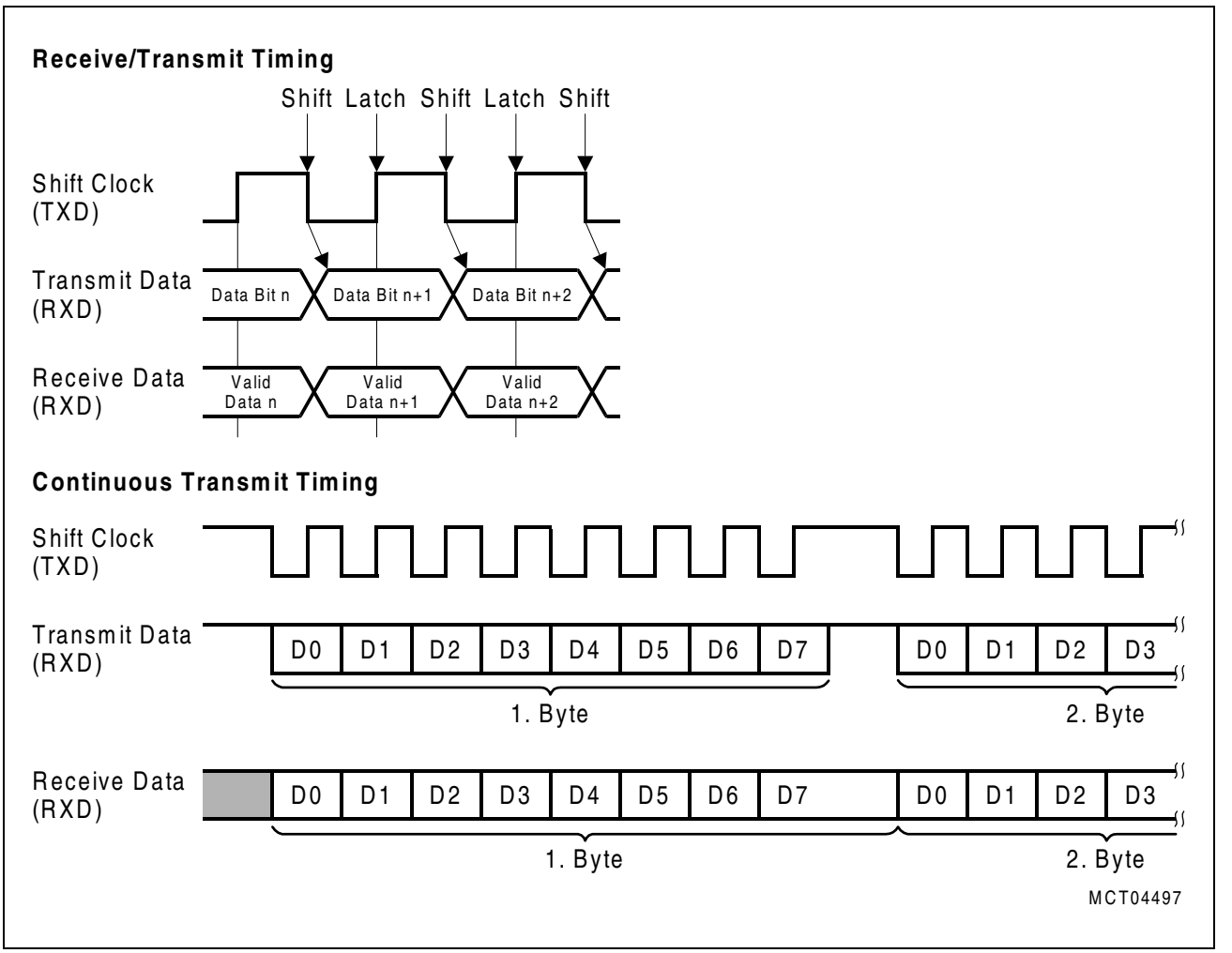


Figure 23-12 ASC Synchronous Mode Waveforms

23.1.5 Baud Rate Generation

The serial channel ASC has its own dedicated 13-bit baud-rate generator with reload capability, allowing baud rate generation independent of other timers.

The baud-rate generator is clocked with a clock (f_{DIV}) derived via a prescaler from the ASC input clock f_{ASC}. The baud rate timer is counting downwards and can be started or stopped through the baud-rate generator run bit CON.R. Each underflow of the timer provides one clock pulse to the serial channel. The timer is reloaded with the value stored in its 13-bit reload register each time it underflows. The resulting clock f_{BRT} is again divided by a factor for the baud rate clock (±16 in asynchronous modes and ±4 in synchronous mode). The prescaler is selected by the bits CON.BRS and CON.FDE. In addition to the two fixed dividers, a fractional divider prescaler unit is available in the Asynchronous Modes that allows selection of prescaler divider ratios of n/512 with n = 0 - 511. Therefore, the baud rate of ASC is determined by the module clock, the content of FDV, the reload value of BG, and the operating mode (asynchronous or synchronous).

Register BG is the dual-function Baud-rate Generator/Reload register. Reading BG returns the contents of the timer BR_VALUE (bits 15 … 13 return zero), while writing to BG always updates the reload register (bits 15 … 13 are insignificant).

An auto-reload of the timer with the contents of the reload register is performed each time BG is written to. However, if CON.R = 0 at the time the write operation to BG is performed, the timer will not be reloaded until the first instruction cycle after CON.R = 1. For a clean baud rate initialization BG should only be written if CON.R = 0. If BG is written with CON.R = 1, an unpredicted behavior of the ASC may occur during running transmit or receive operations.

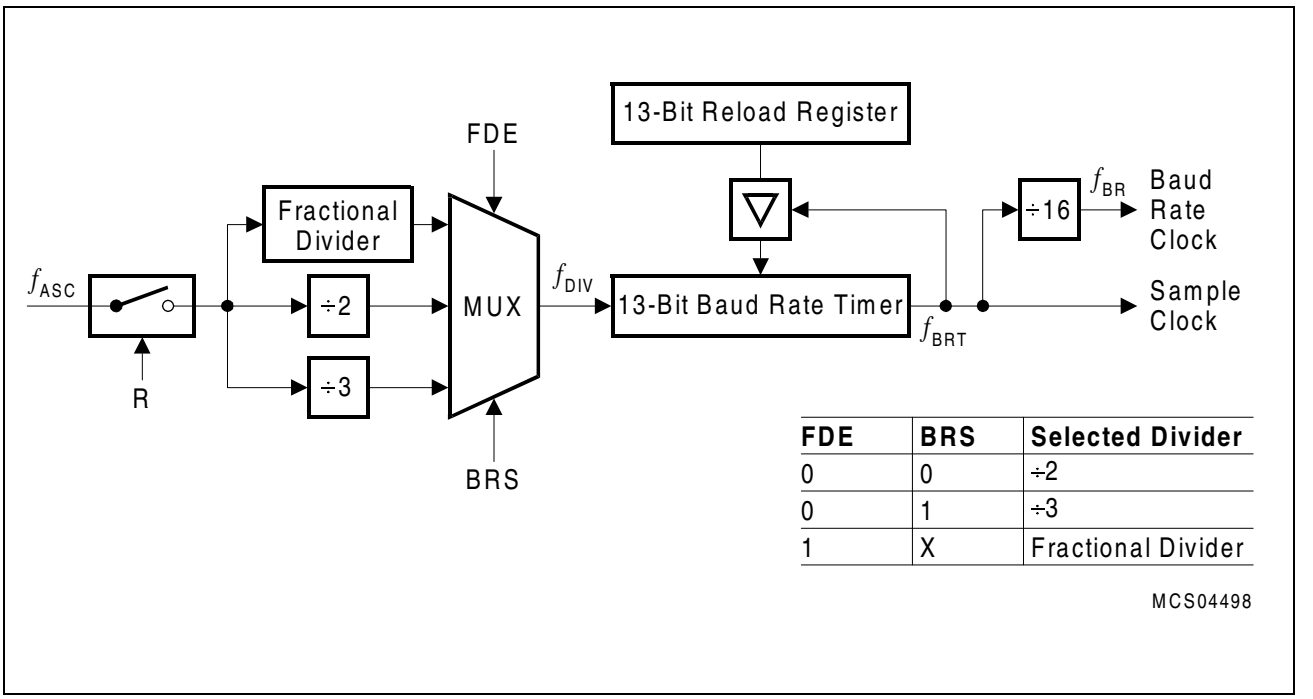
23.1.5.1 Baud Rate in Asynchronous Mode

For asynchronous operation, the baud-rate generator provides a clock f_{BRT} with sixteen times the rate of the established baud rate. Every received bit is sampled at the 7th, 8th and 9th cycle of this clock. The clock divider circuitry, which generates the input clock for the 13-bit baud rate timer, is extended by a fractional divider circuitry that allows the adjustment of more accurate baud rates and the extension of the baud rate range.

The baud rate of the baud-rate generator depends on the settings of the following bits and register values:

- Input clock f_{ASC}
- Selection of the baud rate timer input clock f_{DIV} by bits CON.FDE and CON.BRS
- If bit CON.FDE = 1 (fractional divider): value of register FDV
- Value of the 13-bit reload register BG

The output clock of the baud rate timer with the reload register is the sample clock in the asynchronous modes of the ASC. For baud rate calculations, this baud rate clock f_{BR} is derived from the sample clock f_{BRT} by a division by sixteen.



FDE	BRS	Selected Divider
0	0	÷2
0	1	÷3
1	X	Fractional Divider

Figure 23-13 ASC Baud-rate Generator Circuitry in Asynchronous Modes

Using the Fixed Input Clock Divider

The baud rate for asynchronous operation of the serial channel ASC when using the fixed input clock divider ratios (CON.FDE = 0) and the required reload value for a given baud rate can be determined by the following formulas:

Table 23-3 Asynchronous Baud Rate Formulas using the Fixed Input Clock Dividers

FDE	BRS	BG	Formula
0	0	0 … 8191	$\text{Baudrate} = \frac{f_{ASC}}{32 \times (BG + 1)}$ $BG = \frac{f_{ASC}}{32 \times \text{Baudrate}} - 1$
	1		$\text{Baudrate} = \frac{f_{ASC}}{48 \times (BG + 1)}$ $BG = \frac{f_{ASC}}{48 \times \text{Baudrate}} - 1$

BG represents the content of the reload register BG (BR_VALUE), taken as unsigned 13-bit integer.

The maximum baud rate that can be achieved for the asynchronous modes when using the two fixed clock dividers and a module clock of 75 MHz is 2.34375 MBaud. **Table 23-4** lists various commonly used baud rates together with the required reload values and the deviation errors compared to the intended baud rate.

Table 23-4 Typical Asynchronous Baud Rates using Fixed Input Clock Dividers

Baud Rate	BRS = 0, f_{ASC} = 75 MHz		BRS = 1, f_{ASC} = 75 MHz	
	Deviation Error	Reload Value	Deviation Error	Reload Value
2.34375 MBaud	–	0000_H	NA	NA
1.5625 MBaud	–	–	NA	0000_H
19.2 kBaud	+0.0%	0079_H	+0.5% / -0.7%	0050_H / 0051_H
9600 Baud	+0.0% / -0.3%	$00F3_H$ / $00F4_H$	+0.4% / -0.1%	$00A1_H$ / $00A2_H$
4800 Baud	+0.0% / -0.1%	$01E7_H$ / $01E8_H$	+0.1% / -0.1%	0144_H / 0145_H
2400 Baud	+0.0% / -0.0%	$03CF_H$ / $03D0_H$	+0.0% / -0.1%	$028B_H$ / $028C_H$
1200 Baud	+0.0% / -0.5%	$07A0_H$ / $07A1_H$	+0.0% / -0.0%	0515_H / 0516_H

Note: CON.FDE must be 0 to achieve the baud rates in the table above. The deviation errors given in the table above are rounded. Using a baud rate crystal will provide correct baud rates without deviation errors.

Using the Fractional Divider

When the fractional divider is selected, the input clock f_{DIV} for the baud rate timer is derived from the module clock f_{ASC} by a programmable divider. If CON.FDE = 1, the fractional divider is activated. It divides f_{ASC} by a fraction of n/512 for any value of n from 0 to 511. If n = 0, the divider ratio is 1, which means that $f_{DIV} = f_{ASC}$. In general, the fractional divider allows the baud rate to be programmed with a much better accuracy than with the two fixed prescaler divider stages.

Table 23-5 Asynchronous Baud Rate Formulas using the Fractional Input Clock Divider

FDE	BRS	BG	FDV	Formula
1	–	0 … 8191	1 … 511	$\text{Baudrate} = \frac{FDV}{512} \times \frac{f_{ASC}}{16 \times (BG + 1)}$
			0	$\text{Baudrate} = \frac{f_{ASC}}{16 \times (BG + 1)}$

Note: BG represents the contents of the reload register BG (BR_VALUE), taken as an unsigned 13-bit integer. FDV represents the contents of the fractional divider register (FD_VALUE) taken as an unsigned 9-bit integer.

Table 23-6 Typical Asynchronous Baud Rates using the Fractional Input Clock Divider

f_{ASC}	Desired Baud Rate	BG	FDV	Resulting Baud Rate	Deviation
25 MHz	115.2 kBaud	7	302	115.204 kBaud	< 0.01%
	57.6 kBaud	15	302	57.602 kBaud	< 0.01%
	38.4 kBaud	23	302	39.401 kBaud	< 0.01%
	19.2 kBaud	47	302	19.201 kBaud	< 0.01%
50 MHz	115.2 kBaud	15	302	115.203 kBaud	0.00%
	57.6 kBaud	15	151	57.601 kBaud	0.00%
	38.4 kBaud	23	151	38.401 kBaud	0.00%
	19.2 kBaud	47	151	19.200 kBaud	0.00%
75 MHz	115.2 kBaud	23	302	115.204 kBaud	< 0.01%
	57.6 kBaud	47	302	57.602 kBaud	< 0.01%
	38.4 kBaud	71	302	39.401 kBaud	< 0.01%
	19.2 kBaud	95	302	19.201 kBaud	< 0.01%

23.1.5.2 Baud Rate in Synchronous Mode

For synchronous operation, the baud-rate generator provides a clock with four times the rate of the established baud rate (see **Figure 23-14**).

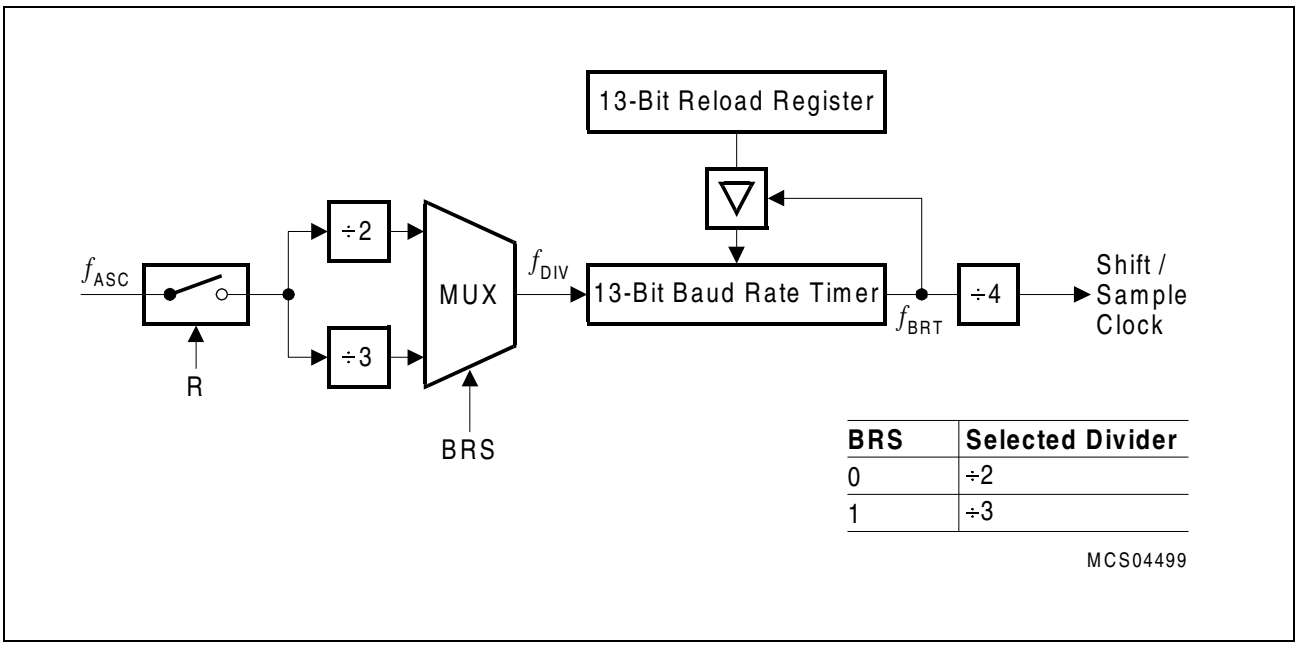


Figure 23-14 ASC Baud-rate Generator Circuitry in Synchronous Mode

The baud rate for synchronous operation of serial channel ASC can be determined by the formulas as shown in **Table 23-7**.

Table 23-7 Synchronous Baud Rate Formulas

BRS	BG	Formula
0	0 … 8191	$\text{Baudrate} = \frac{f_{ASC}}{8 \times (BG + 1)}$ $\quad BG = \frac{f_{ASC}}{8 \times \text{Baudrate}} - 1$
1		$\text{Baudrate} = \frac{f_{ASC}}{12 \times (BG + 1)}$ $\quad BG = \frac{f_{ASC}}{12 \times \text{Baudrate}} - 1$

Note: BG represents the contents of the reload register (BR_VALUE), taken as unsigned 13-bit integers.

The maximum baud rate that can be achieved in Synchronous Mode when using a module clock of 75 MHz is 9.375 MBaud.

23.1.6 Hardware Error Detection Capabilities

To improve the safety of serial data exchange, the serial channel ASC provides an error interrupt request flag that indicates the presence of an error and three (selectable) error status flags in register CON that indicate which error has been detected during reception. Upon completion of a reception, the error interrupt request line EIR will be activated

simultaneously with the receive interrupt request line RIR, if one or more of the following conditions are met:

- If the framing error detection enable bit CON.FEN is set and any of the expected stop bits is not high, the framing error flag CON.FE is set, indicating that the error interrupt request is due to a framing error (Asynchronous Mode only).
- If the parity error detection enable bit CON.PEN is set in the modes where a parity bit is received and the parity check on the received data bits proves false, the parity error flag CON.PE is set, indicating that the error interrupt request is due to a parity error (Asynchronous Mode only).
- If the overrun error detection enable bit CON.OEN is set and the last character received was not read out of the receive buffer by software or DMA transfer at the time the reception of a new frame is complete, the overrun error flag CON.OE is set indicating that the error interrupt request is due to an overrun error (asynchronous and synchronous mode).

23.1.7 Interrupts

Four interrupt sources are provided for serial channel ASC. Line TIR indicates a transmit interrupt, TBIR indicates a transmit buffer interrupt, RIR indicates a receive interrupt, and EIR indicates an error interrupt of the serial channel. The interrupt output lines TBIR, TIR, RIR, and EIR are activated (active state) for two periods of the module clock f_{ASC}. The interrupt control unit provides interrupt request flags that are set when these interrupt output lines are activated.

The cause of an error interrupt request EIR (framing, parity, overrun error) can be identified by the error status flags FE, PE, and OE located in control register CON.

Note: In contrary to the error interrupt request line EIR, the error status flags FE/PE/OE are not reset automatically but must be cleared by software.

For normal operation (that is, other than error interrupt) the ASC provides three interrupt requests to control data exchange via this serial channel:

- TBIR is activated when data is moved from TBUF to the transmit shift register.
- TIR is activated before the last bit of an asynchronous frame is transmitted, or after the last bit of a synchronous frame has been transmitted.
- RIR is activated when the received frame is moved to RBUF.

Note: While the receive task is handled by a single interrupt handler, the transmitter is serviced by two interrupt handlers. This provides advantages for the servicing software.

For single transfers it is sufficient to use the transmitter interrupt (TIR), which indicates that the previously loaded data has been transmitted, except for the last bit of an asynchronous frame.

For multiple back-to-back transfers it is necessary to load the following piece of data at last until the time the last bit of the previous frame has been transmitted. In Asynchronous Mode, this leaves just one bit-time for the handler to respond to the transmitter interrupt request; in Synchronous Mode, it is entirely impossible.

Using the Transmit Buffer Interrupt (TBIR) to reload transmit data gives the time to transmit a complete frame for the service routine, as TBUF may be reloaded while the previous data is still being transmitted.

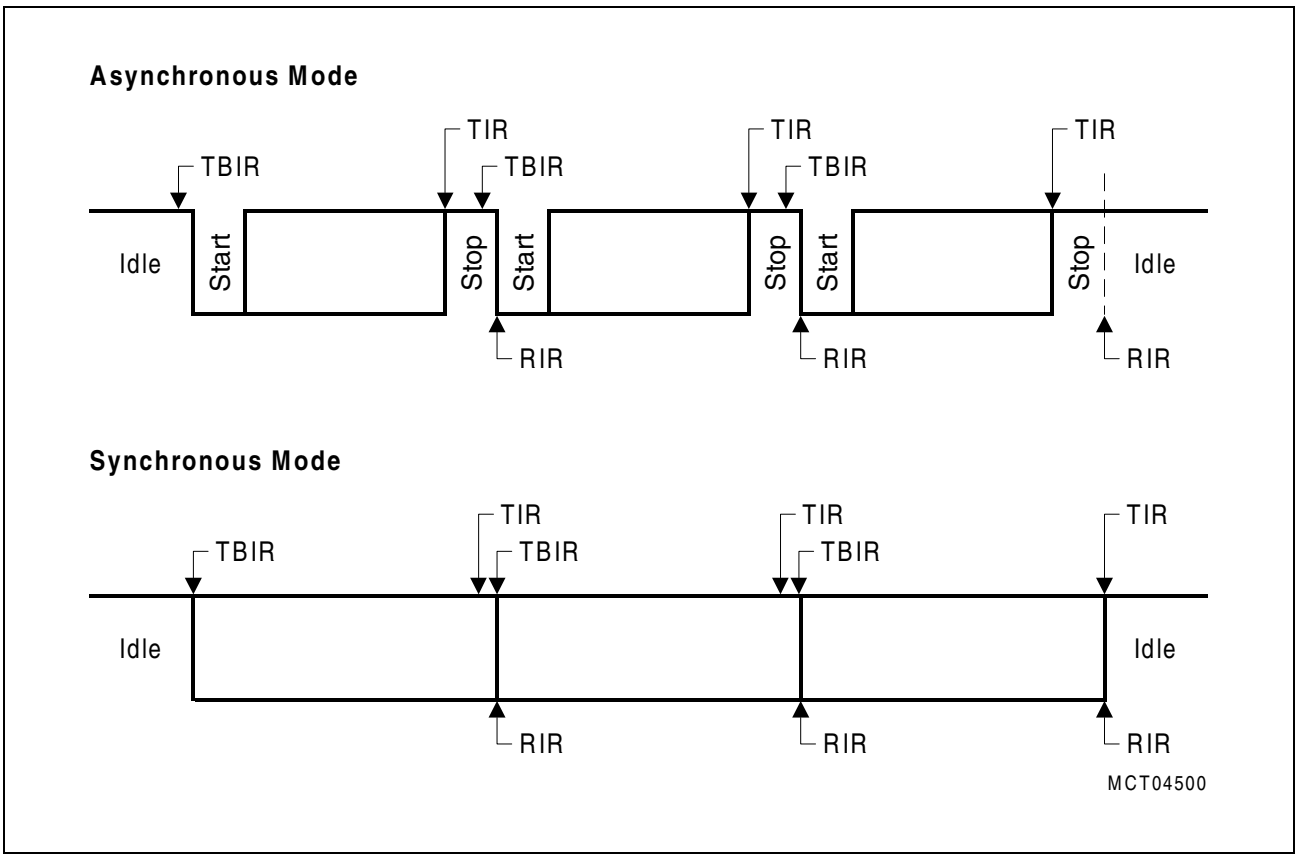


Figure 23-15 ASC Interrupt Generation

As shown in **Figure 23-15** above, TBIR is an early trigger for the reload routine, while TIR indicates the completed transmission. Therefore, software using handshake should rely on TIR at the end of a data block to ensure that all data has actually been transmitted.

23.2 ASC Kernel Registers

Figure 23-16 and Table 23-8 show all registers associated with the ASC Kernel.

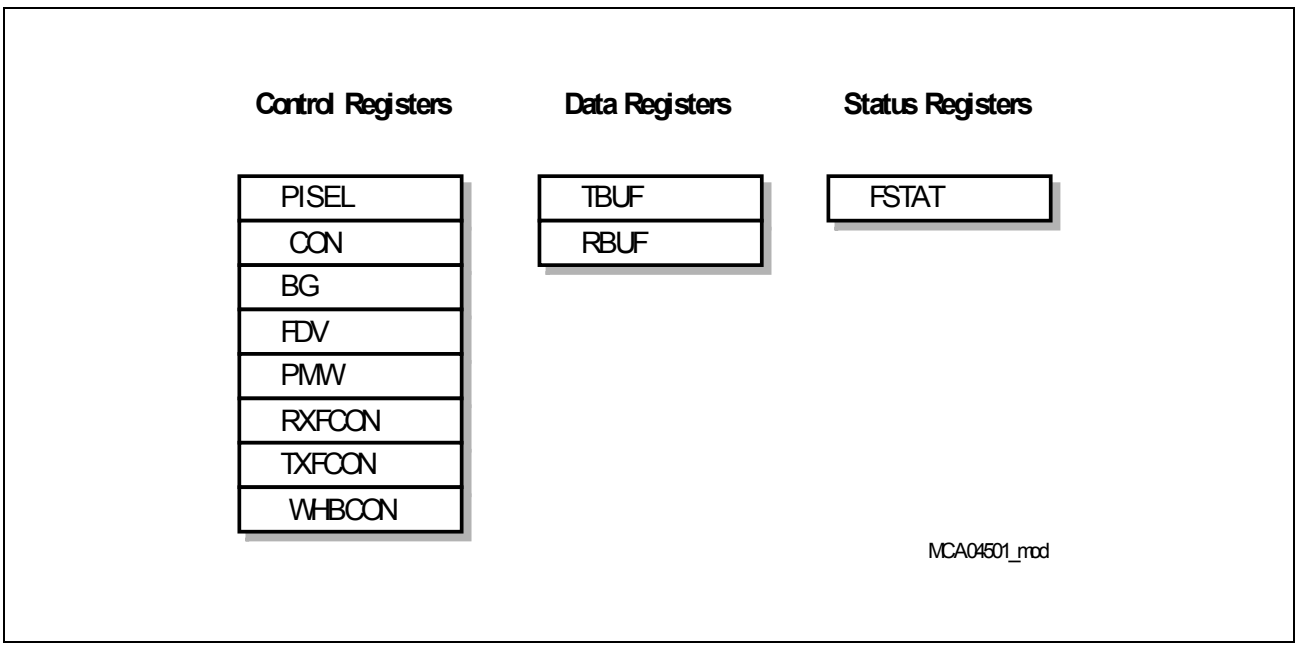


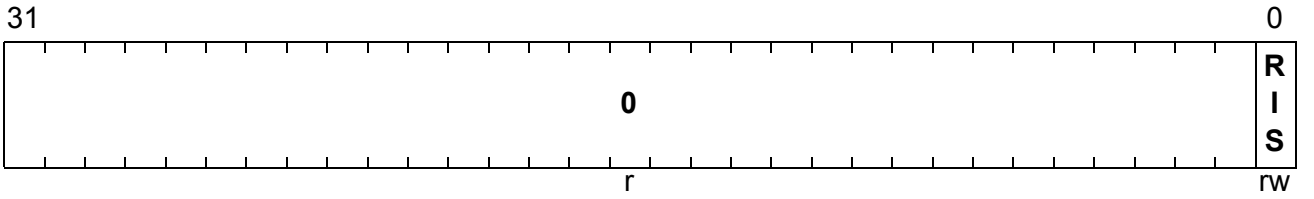
Figure 23-16 ASC Kernel Registers

Table 23-8 ASC Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
PISEL	Peripheral Input Select Register	0004_H	Page 23-30
CON	Control Register	0010_H	Page 23-30
BG	Baud Rate Timer Reload Register	0014_H	Page 23-34
FDV	Fractional Divider Register	0018_H	Page 23-35
PMW	IrDA Pulse Mode and Width Register	$001C_H$	Page 23-36
TBUF	Transmit Buffer Register	0020_H	Page 23-36
RBUF	Receive Buffer Register	0024_H	Page 23-37
RXFCON	Receive FIFO Control Register	0040_H	Page 23-38
TXFCON	Transmit FIFO Control Register	0044_H	Page 23-40
FSTAT	FIFO Status Register	0048_H	Page 23-41
WHBCON	Write Hardware Bits Control Register	0050_H	Page 23-33

The PISEL register controls the input signal selection of the ASC module. Each input of the module kernel receiver signals has associated two input lines.

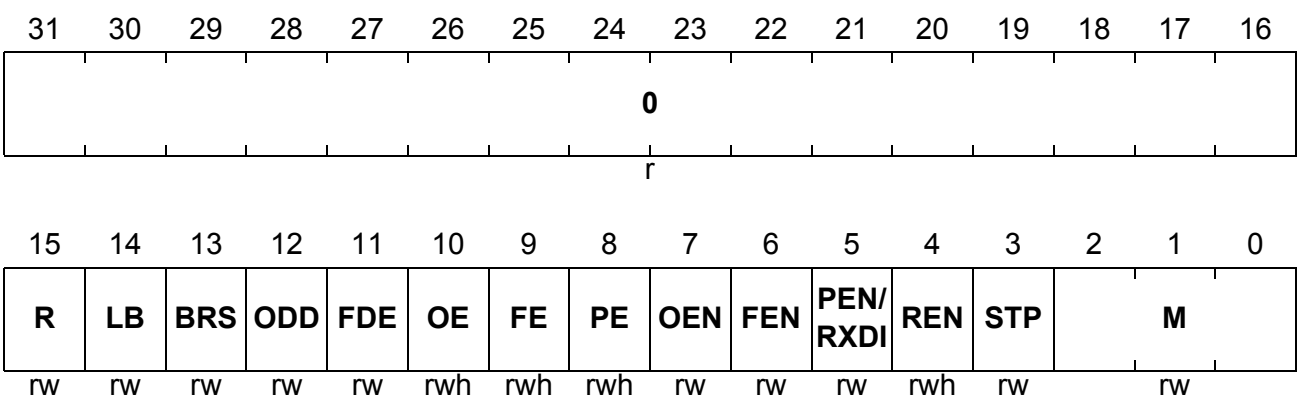
PISEL
Peripheral Input Select Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RIS	0	rw	**Receive Input Select** 0 Default input port is selected for receiver input 1 Alternate input port is selected for receiver input
0	[31:1]	0	**Reserved;** read as 0; should be written with 0.

The serial operating modes of the ASC module are controlled by its control register CON. This register contains control bits for mode and error check selection, and status flags for error identification.

CON
Control Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
M	[2:0]	rw	**Mode Selection** 000 8-bit data Synchronous Mode 001 8-bit data Asynchronous Mode 010 IrDA mode, 8-bit data Asynchronous Mode 011 7-bit data + parity Asynchronous Mode 100 9-bit data Asynchronous Mode 101 8-bit data + wake-up bit Asynchronous Mode 110 Reserved. Do not use this combination! 111 8-bit data + parity Asynchronous Mode
STP	3	rw	**Number of Stop Bit Selection** 0 One stop bit 1 Two stop bits
REN	4	rwh	**Receiver Enable Control** 0 Receiver disabled 1 Receiver enabled *Note: Bit is reset by hardware after reception of byte in Synchronous Mode.*
PEN/ RXDI	5	rw	**Parity Check Enable/ RXD Input Inverter Enable in IrDA Mode** All Asynchronous modes except IrDA mode 0 Ignore parity 1 Check parity Only IrDA mode (M = 010_B) 0 RXD input is not inverted 1 RXD input is inverted
FEN	6	rw	**Framing Check Enable (asynchronous modes only)** 0 Ignore framing errors 1 Check framing errors
OEN	7	rw	**Overrun Check Enable** 0 Ignore overrun errors 1 Check overrun errors
PE	8	rwh	**Parity Error Flag** Set by hardware on a parity error (PEN = 1). Must be reset by software.
FE	9	rwh	**Framing Error Flag** Set by hardware on a framing error (FEN = 1). Must be reset by software.

Field	Bits	Type	Description
OE	10	rwh	**Overrun Error Flag** Set by hardware on an overrun/underflow error (OEN = 1). Must be reset by software.
FDE	11	rw	**Fractional Divider Enable** 0 Fractional divider disabled 1 Fractional divider is enabled and used as prescaler for baud rate timer (bit BRS is don't care)
ODD	12	rw	**Parity Selection** 0 Even parity selected (parity bit set on odd number of 1s in data) 1 Odd parity selected (parity bit set on even number of 1s in data)
BRS	13	rw	**Baud Rate Selection** 0 Baud rate timer prescaler divide-by-2 selected 1 Baud rate timer prescaler divide-by-3 selected *Note: BRS is don't care if FDE = 1 (fractional divider enabled)*
LB	14	rw	**Loopback Mode Enable** 0 Loop-Back mode disabled 1 Loop-Back mode enabled
R	15	rw	**Baud-rate Generator Run Control** 0 Baud-rate generator disabled (ASC inactive) 1 Baud-rate generator enabled *Note: BR_VALUE should only be written if R = 0.*
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

Note: Serial data transmission or reception is possible only when the run bit CON.R is set to 1. Otherwise, the serial interface is idle. Do not program the mode control field CON.M to the reserved combination to avoid unpredictable behavior of the serial interface.

Critical "rwh" Bits

Register CON contains three error flags: PE, FE and OE. If the software wants to modify only one of these error flags, it uses typically a Read-Modify-Write (RMW) instruction. When one of the other error flags, which is not intended to be modified by the RWM instruction, is changed by hardware after the read access but before write back access of the RMW instruction, it is overwritten with the old bit value and the hardware change

of the bit gets lost. This problem does not affect the bits which are intended to be modified by the RMW instruction. It only affects bits which are not intended to be changed with the RMW instruction.

The three error flags in register CON and the REN bit can be additionally set or reset by software via register WHBCON. This capability avoids the problem with the CON register RMW instruction access to the error flags. WHBCON is a write-only register. Reading WHBCON always returns 0000 0000_H.

WHBCON
Write Hardware Bits Control Register **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

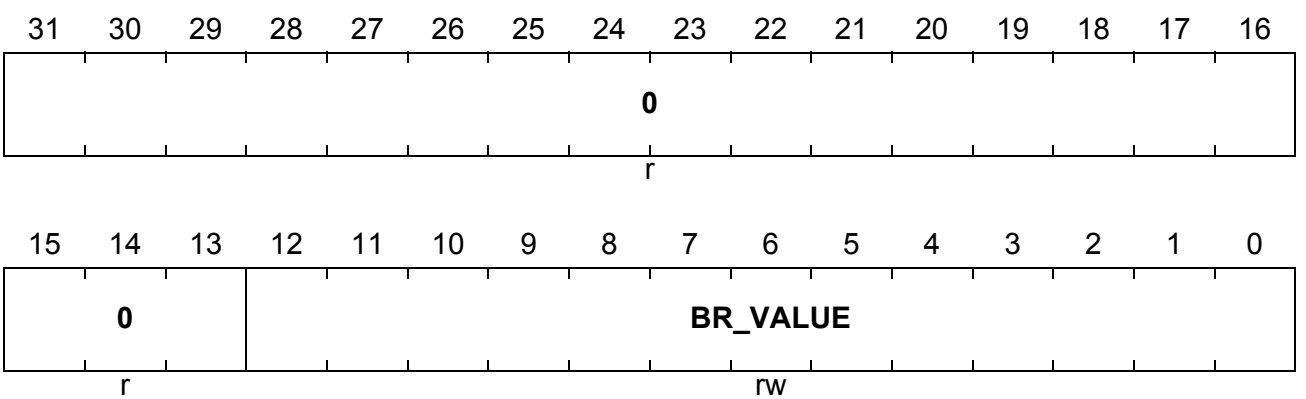
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		SETOE	SETFE	SETPE	CLROE	CLRFE	CLRPE	0		SETREN	CLRREN	0			
r		w	w	w	w	w	w	r		w	w	r			

Field	Bits	Type	Description
CLRREN	4	w	**Clear Receiver Enable Bit** 0 No effect 1 Bit CON.REN is cleared Bit is always read as 0.
SETREN	5	w	**Set Receiver Enable Bit** 0 No effect 1 Bit CON.REN is set Bit is always read as 0.
CLRPE	8	w	**Clear Parity Error Flag** 0 No effect 1 Bit CON.PE is cleared Bit is always read as 0.
CLRFE	9	w	**Clear Framing Error Flag** 0 No effect 1 Bit CON.FE is cleared Bit is always read as 0.

Field	Bits	Type	Description
CLROE	10	w	**Clear Overrun Error Flag** 0 No effect 1 Bit CON.OE is cleared Bit is always read as 0.
SETPE	11	w	**Set Parity Error Flag** 0 No effect 1 Bit CON.PE is set Bit is always read as 0.
SETFE	12	w	**Set Framing Error Flag** 0 No effect 1 Bit CON.FE is set Bit is always read as 0.
SETOE	13	w	**Set Overrun Error Flag** 0 No effect 1 Bit CON.OE is set Bit is always read as 0.
0	[3:0], [7:6], [31:14]	r	**Reserved;** read as 0; should be written with 0.

The baud rate timer reload register BG of the ASC module contains the 13-bit reload value for the baud rate timer in Asynchronous and Synchronous Mode.

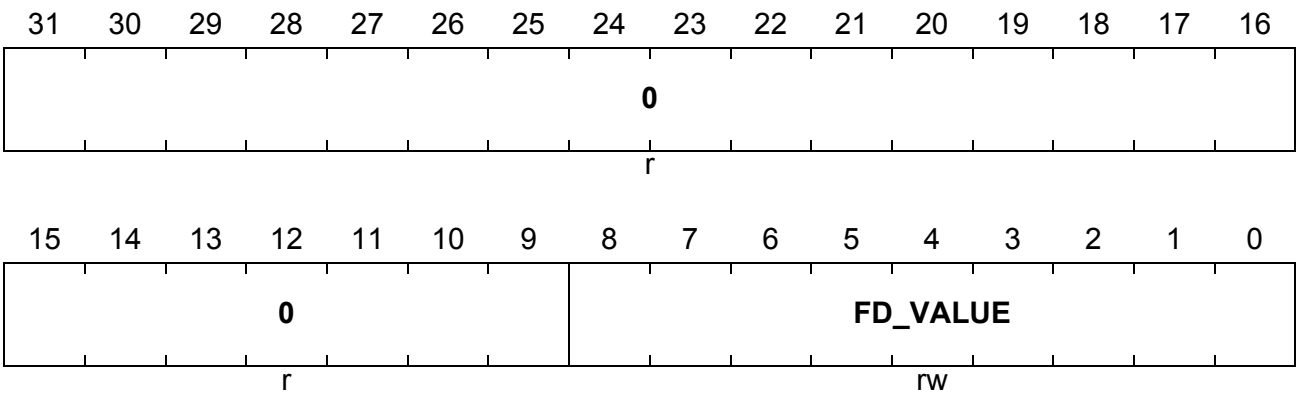
BG
Baud Rate Timer/Reload Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
BR_VALUE	[12:0]	rw	**Baud Rate Timer/Reload Value** Reading returns the 13-bit content of the baud rate timer; writing loads the baud rate timer/reload value. *Note: BG should only be written if R = 0.*
0	[31:13]	r	**Reserved;** read as 0; should be written with 0.

The fractional divider register FDV of the ASC module contains the 9-bit divider value for the fractional divider (asynchronous mode only).

FDV
Fractional Divider Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
FD_VALUE	[8:0]	rw	**Fractional Divider Register Value** FD_VALUE contains the 9-bit value of the fractional divider which defines the fractional divider ratio n/512 (n = 0-511). With n = 0, the fractional divider is switched off (input = output frequency).
0	[31:9]	r	**Reserved;** read as 0; should be written with 0.

The IrDA pulse mode and width register PMW of the ASC module contains the 8-bit IrDA pulse width value and the IrDA pulse width mode select bit. This register is only required in the IrDA operating mode.

PMW
IrDA Pulse Mode/Width Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							IRPW	PW_VALUE							
r							rw	rw							

Field	Bits	Type	Description
PW_VALUE	[7:0]	rw	**IrDA Pulse Width Value** PW_VALUE is the 8-bit value n, which defines the variable pulse width of an IrDA pulse. Depending on the ASC input frequency f_{ASC}, this value can be used to adjust the IrDA pulse width to value which is not equal 3/16 bit time (e.g. 1.6 µs).
IRPW	8	rw	**IrDA Pulse Width Mode Control** 0 IrDA pulse width is 3/16 of the bit time 1 IrDA pulse width is defined by PW_VALUE
0	[31:9]	r	**Reserved;** read as 0; should be written with 0.

The transmitter buffer register TBUF of the ASC module contains the transmit data value in Asynchronous and Synchronous Modes.

TBUF
Transmit Buffer Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

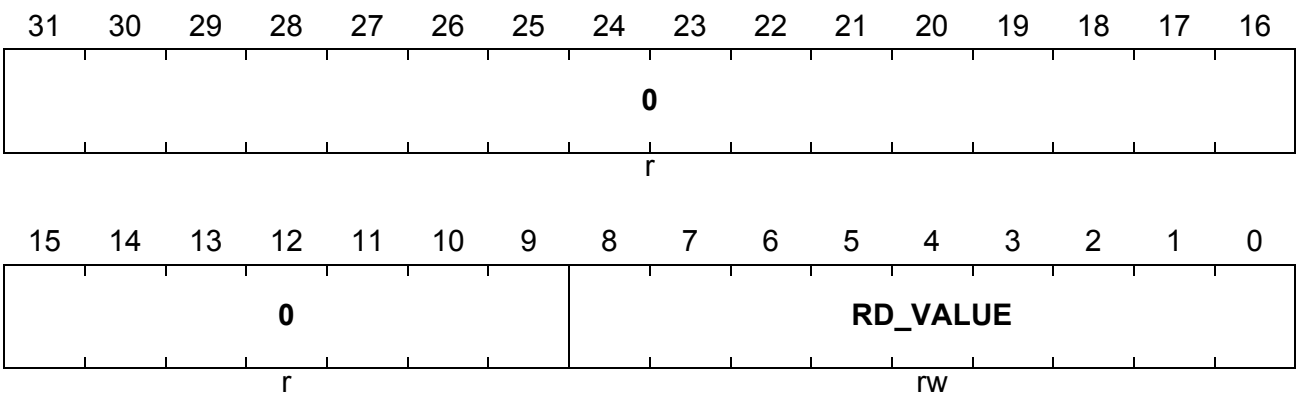
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							TD_VALUE								
r							rw								

Field	Bits	Type	Description
TD_VALUE	[8:0]	rw	**Transmit Data Register Value** TBUF contains the data to be transmitted in asynchronous and synchronous operating mode of the ASC. Data transmission is double buffered; therefore, a new value can be written to TBUF before the transmission of the previous value is complete.
0	[31:9]	r	**Reserved;** read as 0; should be written with 0.

Note: In IrDA Mode, after CON register is set, some delay should be inserted before writing the transmitted data to TBUF register. This is because it takes time to generate the baud rate to transmit the IrDA frame.

The receiver buffer register RBUF of the ASC module contains the receive data value in Asynchronous and Synchronous Modes.

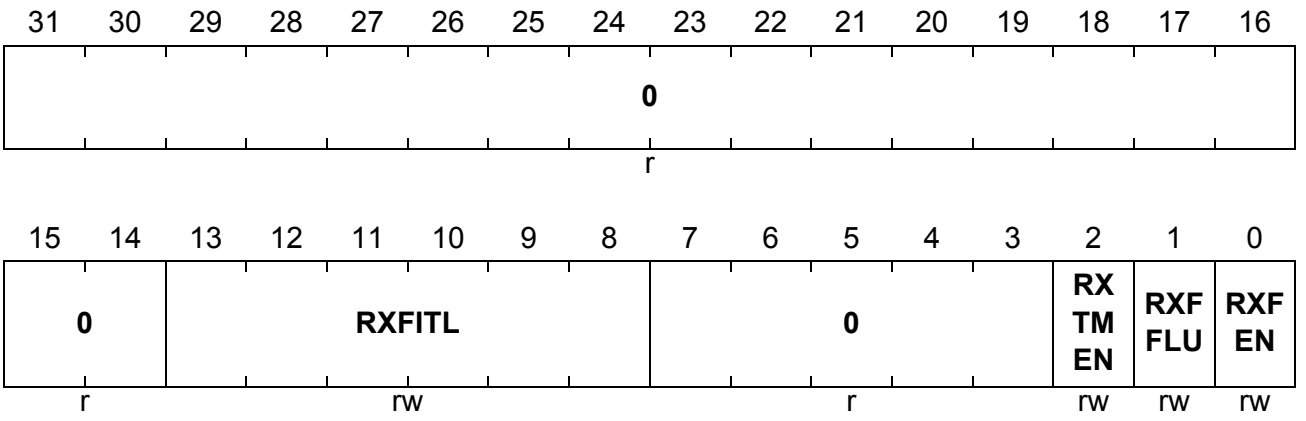
RBUF
Receive Buffer Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RD_VALUE	[8:0]	rw	**Receive Data Register Value** RBUF contains the received data bits and, depending on the selected mode, the parity bit in the asynchronous and synchronous operating modes of the ASC. In Asynchronous Mode, with CON.M = 011_B (7-bit data + parity), the received parity bit is written into RBUF.7. In Asynchronous Mode with CON.M = 111_B (8-bit data + parity), the received parity bit is written into RBUF.8.
0	[31:9]	r	**Reserved;** read as 0; should be written with 0.

The receive FIFO control register RXFIFO contains control bits and bit fields that define the operating mode of the receive FIFO.

RXFCON
Receive FIFO Control Register **Reset Value: 0000 0100$_H$**

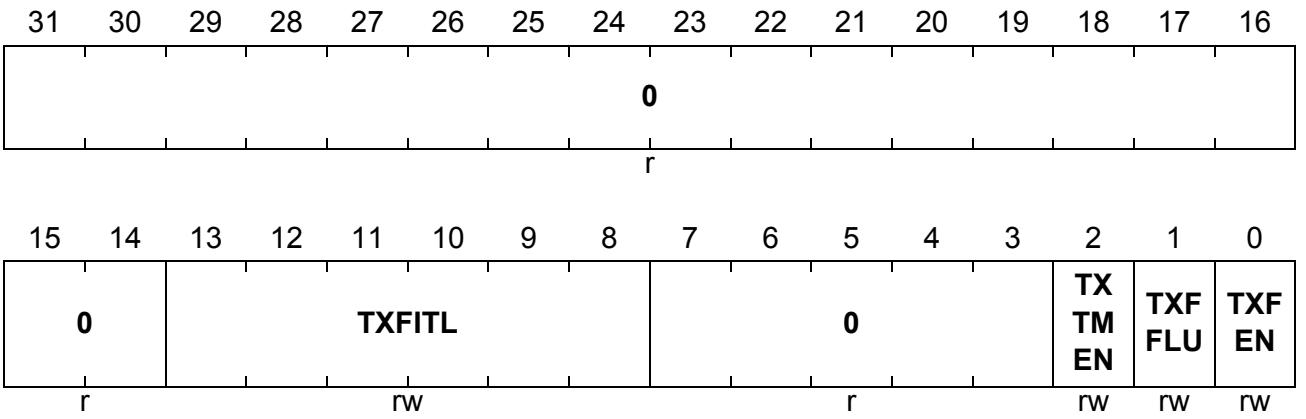


Field	Bits	Type	Description
RXFEN	0	rw	**Receive FIFO Enable** 0 Receive FIFO is disabled 1 Receive FIFO is enabled *Note: Resetting RXFEN automatically flushes the receive FIFO.*
RXFFLU	1	rw	**Receive FIFO Flush** 0 No operation 1 Receive FIFO is flushed *Note: Setting RXFFLU clears bit field FSTAT. RXFFL. Bit RXFFLU is always read as 0.*
RXTMEN	2	rw	**Receive FIFO Transparent Mode Enable** 0 Receive FIFO Transparent Mode is disabled 1 Receive FIFO Transparent Mode is enabled *Note: This bit is not applicable if the receive FIFO is disabled (RXFEN = 0).*

Field	Bits	Type	Description
RXFITL	[13:8]	rw	**Receive FIFO Interrupt Trigger Level** Defines a receive FIFO interrupt trigger level. A receive interrupt request (RIR) is always generated after the reception of a byte when the filling level of the receive FIFO is equal to or greater than RXFITL. 000000 Reserved 000001 Interrupt trigger level is set to one 000010 Interrupt trigger level is set to two … 001000 Interrupt trigger level is set to eight Others Reserved *Note: In Transparent Mode, this bit field is not applicable.* *Note: Combinations defining an interrupt trigger level greater than the configured FIFO size should not be used.*
0	[7:3], [31:14]	r	**Reserved;** read as 0; should be written with 0.

The transmit FIFO control register TXFIFO contains control bits and bit fields that define the operating mode of the transmit FIFO.

TXFCON
Transmit FIFO Control Register **Reset Value: 0000 0100$_H$**

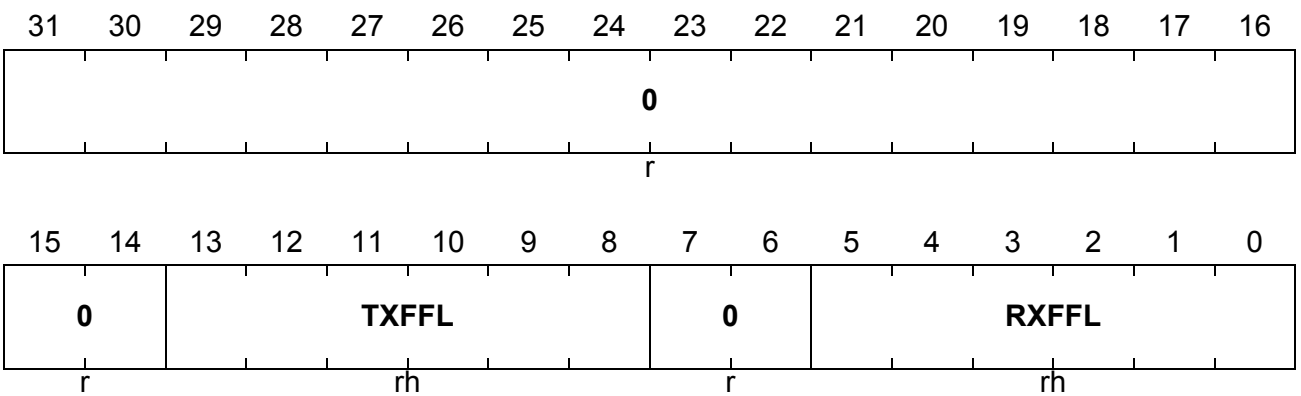


Field	Bits	Type	Description
TXFEN	0	rw	**Transmit FIFO Enable** 0 Transmit FIFO is disabled 1 Transmit FIFO is enabled *Note: Resetting TXFEN automatically flushes the transmit FIFO.*
TXFFLU	1	rw	**Transmit FIFO Flush** 0 No operation 1 Transmit FIFO is flushed *Note: Setting TXFFLU clears bit field FSTAT.TXFFL. Bit TXFFLU is always read as 0.*
TXTMEN	2	rw	**Transmit FIFO Transparent Mode Enable** 0 Transmit FIFO Transparent Mode is disabled 1 Transmit FIFO Transparent Mode is enabled *Note: This bit is not applicable if the transmit FIFO is disabled (TXFEN = 0).*

Field	Bits	Type	Description
TXFITL	[13:8]	rw	**Transmit FIFO Interrupt Trigger Level** Defines a transmit FIFO interrupt trigger level. A transmit interrupt request (TIR) is always generated after the transfer of a byte when the filling level of the transmit FIFO is equal to or lower than TXFITL. 000000 Reserved 000001 Interrupt trigger level is set to one 000010 Interrupt trigger level is set to two … 001000 Interrupt trigger level is set to eight Others Reserved *Note: In Transparent Mode, this bit field is not applicable.* *Note: Combinations defining an interrupt trigger level greater than the configured FIFO size should not be used.*
0	[7:3], [31:14]	r	**Reserved;** read as 0; should be written with 0.

The FIFO status register FSTAT indicates the filling levels of the receive and transmit FIFOs.

FSTAT
FIFO Status Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RXFFL	[5:0]	rh	**Receive FIFO Filling Level** 000000 Receive FIFO is filled with zero byte 000001 Receive FIFO is filled with one byte 000010 Receive FIFO is filled with two bytes … 001000 Receive FIFO is filled with eight bytes Others Reserved *Note: RXFFL is cleared after a receive FIFO flush operation.* *Note: Combinations defining an interrupt trigger level greater than the configured FIFO size should not be used.*
TXFFL	[13:8]	rh	**Transmit FIFO Filling Level** 000000 Transmit FIFO is filled with zero byte 000001 Transmit FIFO is filled with one byte 000010 Transmit FIFO is filled with two bytes … 001000 Transmit FIFO is filled with eight bytes Others Reserved *Note: TXFFL is cleared after a transmit FIFO flush operation.* *Note: Combinations defining an interrupt trigger level greater than the configured FIFO size should not be used.*
0	[7:6], [31:14]	r	**Reserved;** read as 0; should be written with 0.

23.3 ASC0/ASC1/ASC2 Module Implementation

This section describes ASC0/ASC1/ASC2 module interfaces with the clock control, port connections, interrupt control, and address decoding.

23.3.1 Interfaces of the ASC Modules

Figure 23-17 shows the TC1130 specific implementation details and interconnections of the ASC0/ASC1/ASC2 modules. The serial I/O lines of these modules can be connected either to Port 0 or to Port 2. Each of the ASC modules is further supplied by clock control, interrupt control, address decoding, and port control logic. Two DMA requests can be generated by each ASC module.

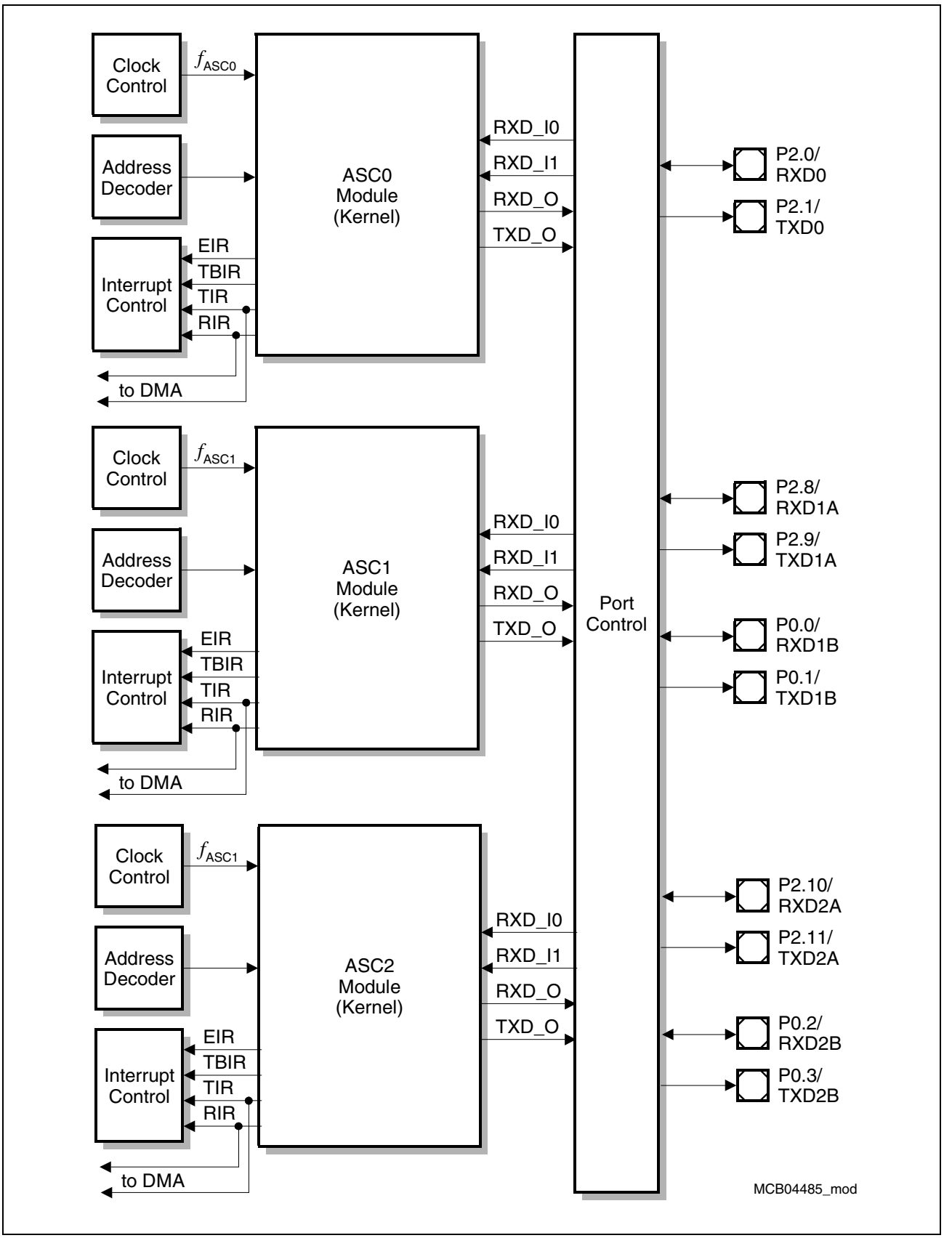


Figure 23-17 ASC0/ASC1/ASC2 Module Implementation and Interconnections

23.3.2 ASC0/ASC1/ASC2 Module Related External Registers

Figure 23-18 summarizes the module related external registers, which are required for ASC0/ASC1/AS2 programming (see also **Figure 23-16** for the module kernel specific registers).

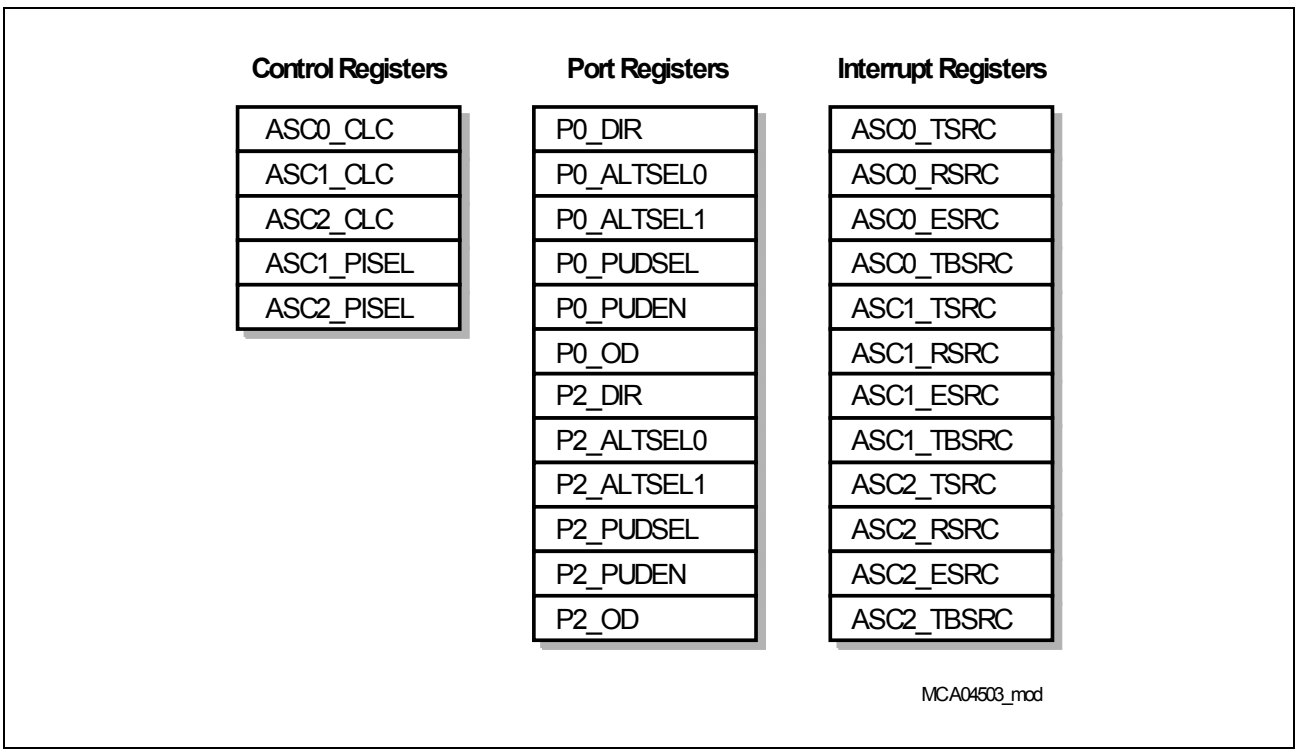
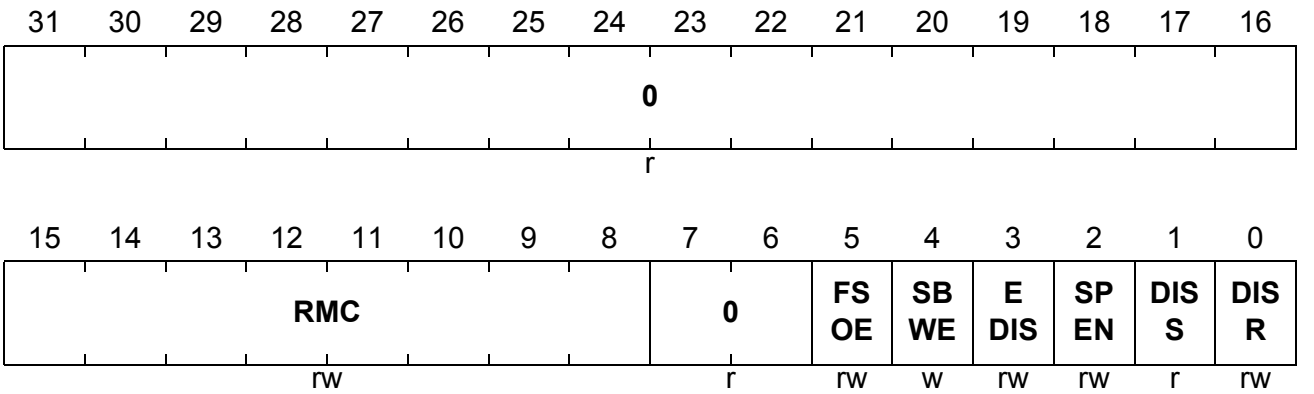


Figure 23-18 ASC0/ASC1/ASC2 Implementation Specific Special Function Registers

23.3.2.1 Clock Control Registers

The clock control register allows the programmer to adapt the functionality and power consumption of an ASC module to the requirements of the application. The table below shows the clock control register functionality which is implemented for the ASC modules. ASC0_CLC is controlling the f_{ASC0} clock signal and ASC1_CLC is controlling the f_{ASC1} clock signal and ASC2_CLC is controlling the f_{ASC2} clock signal.

ASC0_CLC
ASC0 Clock Control Register **Reset Value: 0000 0002$_H$**
ASC1_CLC
ASC1 Clock Control Register **Reset Value: 0000 0002$_H$**
ASC2_CLC
ASC2 Clock Control Register **Reset Value: 0000 0002$_H$**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the module.
SPEN	2	rw	**Module Suspend Enable for OCDS** Used for enabling the suspend mode.
EDIS	3	rw	**External Request Disable** Used for controlling the external clock disable request.
SBWE	4	w	**Module Suspend Bit Write Enable for OCDS** Defines whether SPEN and FSOE are write protected.
FSOE	5	rw	**Fast Switch Off Enable** Used for fast clock switch off in OCDS suspend mode.
RMC	[15:8]	rw	**8-Bit Clock Divider Value in RUN Mode**
0	[7:6], [31:16]	r	**Reserved;** read as 0; should be written with 0.

Note: After a hardware reset operation, the ASC modules are disabled.

Note: As described in the “System Control Unit” chapter under “Module Clock Divider Control” (“destructive read” access), it must be considered that using the RMC bit field in ASC0_CLC, ASC1_CLC and ASC2_CLC may result in a longer read cycle access time on the FPI Buses for the ASC.

23.3.2.2 Peripheral Input Select Register

The ASC1/ASC2 module provides a Peripheral Input Select Register that is used to switch the RXD input lines of the ASC1/ASC2 module kernels to either Port 0 or Port 2 as shown in **Figure 23-19**.

*Note: As shown in **Figure 23-19**, the RXD lines of the ASC modules can also be output lines in Synchronous Mode. Port line input/output switching is controlled by the input/output control registers DIR.*

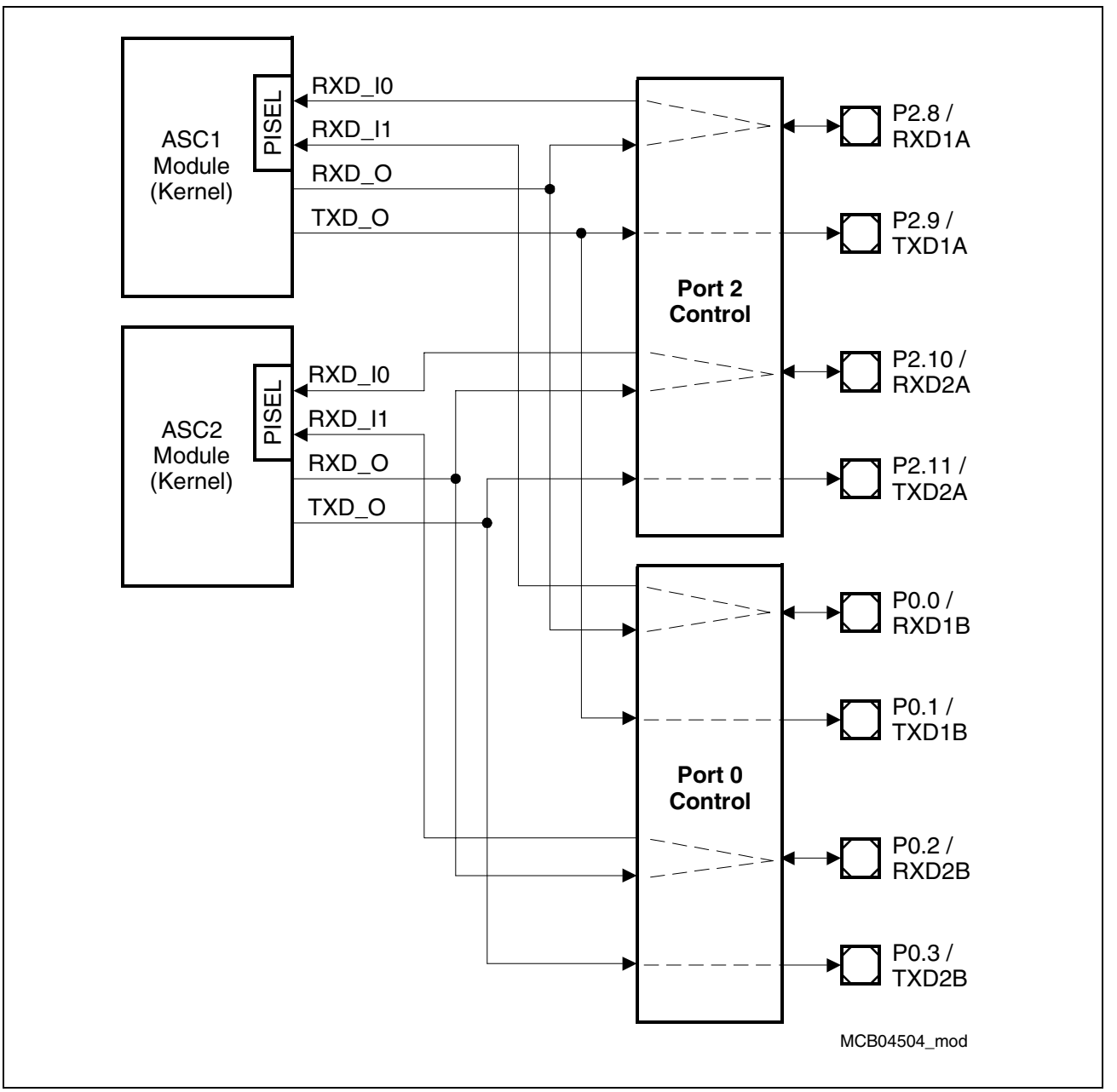
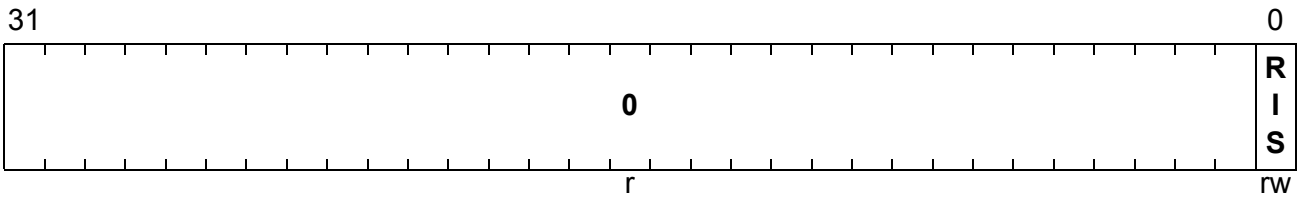


Figure 23-19 RXD Input Line Selection of the ASC Modules

ASC1_PISEL
ASC1 Peripheral Input Select Register

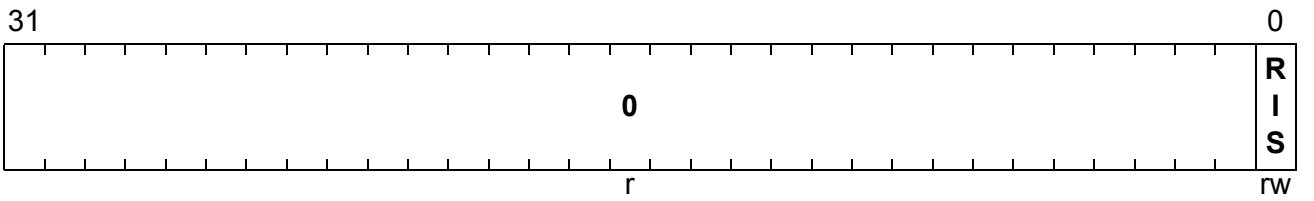
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
RIS	0	rw	**Receive Input Select** 0 ASC1 receiver input RXD1A selected 1 ASC1 receiver input RXD1B selected
0	[31:1]	0	**Reserved;** read as 0; should be written with 0.

ASC2_PISEL
ASC2 Peripheral Input Select Register

Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
RIS	0	rw	**Receive Input Select** 0 ASC2 receiver input RXD1A selected 1 ASC2 receiver input RXD1B selected
0	[31:1]	0	**Reserved;** read as 0; should be written with 0.

23.3.2.3 Port Control

The interconnections between the ASC modules and the port I/O lines are controlled in the port logics. The following port control operation selections must be executed (additionally to the PISEL programming):

- Input/output direction selection (DIR registers)
- Alternate function selection (ALTSEL0 and ALTSEL1 registers)
- Input/Output driver characteristic control (PUDSEL, PUDEN and OD registers)

Input/Output Function Selection

The port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as pull-up/down devices, port direction (input/output), open-drain, and alternate output selections. The I/O lines for the ASC modules are controlled by the port input/output control registers of Port 0 and Port 2.

Table 23-9 shows how bits and bit fields must be programmed for the required I/O functionality of the ASC I/O lines. This table also shows the values of the peripheral input select registers.

Table 23-9 ASC0/ASC1/ASC2 I/O Control Selection and Setup

Module	Port Lines	PISEL Register	Input/Output Control Register Bits	I/O
ASC0	P2.0/RXD0	–	P2_DIR.P0 = 0_B	Input
		–	P2_DIR.P0 = 1_B	Output
			P2_ALTSEL0.P0 = 1_B	
			P2_ALTSEL1.P0 = 0_B	
	P2.1/TXD0	–	P2_DIR.P1 = 1_B	Output
			P2_ALTSEL0.P1 = 1_B	
			P2_ALTSEL1.P1 = 0_B	

Table 23-9 ASC0/ASC1/ASC2 I/O Control Selection and Setup (cont'd)

Module	Port Lines	PISEL Register	Input/Output Control Register Bits	I/O
ASC1	P2.8/RXD1A	ASC1_PISEL.RIS = 0	P2_DIR.P8 = 0_B	Input
		–	P2_DIR.P8 = 1_B	Output
			P2_ALTSEL0.P8 = 1_B	
			P2_ALTSEL1.P8 = 0_B	
	P0.0/RXD1B	ASC1_PISEL.RIS = 1	P0_DIR.P0 = 0_B	Input
		–	P0_DIR.P0 = 1_B	Output
			P0_ALTSEL0.P0 = 0_B	
			P0_ALTSEL1.P0 = 1_B	
	P2.9/TXD1A	–	P2_DIR.P9 = 1_B	Output
			P2_ALTSEL0.P9 = 1_B	
			P2_ALTSEL1.P9 = 0_B	
	P0.1/TXD1B	–	P0_DIR.P1 = 1_B	Output
			P0_ALTSEL0.P1 = 0_B	
			P0_ALTSEL1.P1 = 1_B	
ASC2	P2.10/RXD2A	ASC2_PISEL.RIS = 0	P2_DIR.P10 = 0_B	Input
		–	P2_DIR.P10 = 1_B	Output
			P2_ALTSEL0.P10 = 1_B	
			P2_ALTSEL1.P10 = 0_B	
	P0.2/RXD2B	ASC2_PISEL.RIS = 1	P0_DIR.P2 = 0_B	Input
		–	P0_DIR.P2 = 1_B	Output
			P0_ALTSEL0.P2 = 0_B	
			P0_ALTSEL1.P2 = 1_B	
	P2.11/TXD2A	–	P2_DIR.P11 = 1_B	Output
			P2_ALTSEL0.P11 = 1_B	
			P2_ALTSEL1.P11 = 0_B	
	P0.3/TXD2B	–	P0_DIR.P3 = 1_B	Output
			P0_ALTSEL0.P3 = 0_B	
			P0_ALTSEL1.P3 = 1_B	

*Note: In synchronous operating mode of the ASC, the type of the selected RXD port pin (input or output) is **not** automatically controlled by the ASC but must be defined by a user program by writing the appropriate bit field in the DIR registers.*

P0_DIR
Port 0 Direction Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-3)**	n	rw	**Port 0 Pin 0 - 3 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for ASC I/O port control.

P2_DIR
Port 2 Direction Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-1, 8-11)**	n	rw	**Port 2 Pin 0, 1, 8-11 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for ASC I/O port control.

P0_ALTSELn (n = 0, 1)
Port 0 Alternate Select Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	**P14**	**P13**	**P12**	**P11**	**P10**	**P9**	**P8**	**P7**	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 23-10 Function of the Bits P0_ALTSEL0.Pn and P0_ALTSEL1.Pn (n = 0-3) [1)]

P0_ALTSEL0.Pn	P0_ALTSEL1.Pn	Function
0	1	Alternate Select 2

1) Shaded bits and bit field are don't care for ASC I/O port control.

P2_ALTSELn (n = 1, 0)
Port 2 Alternate Select Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	**P14**	**P13**	**P12**	**P11**	**P10**	**P9**	**P8**	**P7**	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 23-11 Function of the Bits P2_ALTSEL0.Pn and P2_ALTSEL1.Pn (n = 0, 1, 8-11)[1)]

P2_ALTSEL0.Pn	P2_ALTSEL1.Pn	Function
1	0	Alternate Select 1

1) Shaded bits and bit field are don't care for ASC I/O port control.

The ASC ports also offer the possibility to configure the following output characteristics:

- Push/pull (optional pull-up/pull-down)
- Open drain with internal pull-up
- Open drain with external pull-up

P0_PUDSEL
Port 0 Pull-Up/Pull-Down Select Register **Reset Value: 0000 FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 0-3)	n	rw	**Pull-Up/Pull-Down Select Port 0 Bit n[1)]** 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for ASC I/O port control.

P2_PUDSEL
Port 2 Pull-Up/Pull-Down Select Register **Reset Value: 0000 0FFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
r				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0, 1, 8-11)**	n	rw	**Pull-Up/Pull-Down Select Port 2 Bit n**[1)] 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:12]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for ASC I/O port control.

P0_PUDEN
Port 0 Pull-Up/Pull-Down Enable Register **Reset Value: 0000 FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-3)**	n	rw	**Pull-Up/Pull-Down Enable at Port 0 Bit n**[1)] 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for ASC I/O port control.

P2_PUDEN
Port 2 Pull-Up/Pull-Down Enable Register

Reset Value: 0000 0FFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
r				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0, 1, 8-11)**	n	rw	**Pull-Up/Pull-Down Enable at Port 2 Bit n**[1)] 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:12]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for ASC I/O port control.

P0_OD
Port 0 Open Drain Control Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-3)**	n	rw	**Port 0 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for ASC I/O port control.

P2_OD
Port 2 Open Drain Control Register **Reset Value: 0000 F000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	**P14**	**P13**	**P12**	**P11**	**P10**	**P9**	**P8**	**P7**	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0, 1, 8-11)**	n	rw	**Port 2 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for ASC I/O port control.

23.3.2.4 Interrupt Registers

The 12 interrupts of the ASC0 and ASC1 modules are controlled by the following service request control registers:

- ASC0_TSRC, ASC1_TSRC, ASC2_TSRC → controls the transmit interrupts
- ASC0_RSRC, ASC1_RSRC, ASC2_RSRC → controls the receive interrupts
- ASC0_ESRC, ASC1_ESRC, ASC2_ESRC → controls the error interrupts
- ASC0_TBSRC, ASC1_TBSRC, ASC2_TBSRC → controls the transmit buffer empty interrupts

Note: Further details on interrupt handling and processing are described in the chapter "Interrupt System" of the TC1130 System Units User's Manual.

ASC0_TSRC
ASC0 Transmit Interrupt Service Request Control Register
ASC0_RSRC
ASC0 Receive Interrupt Service Request Control Register
ASC0_ESRC
ASC0 Error Interrupt Service Request Control Register
ASC0_TBSRC
ASC0 Transmit Buffer Interrupt Service Request Control Register
ASC1_TSRC
ASC1 Transmit Interrupt Service Request Control Register
ASC1_RSRC
ASC1 Receive Interrupt Service Request Control Register
ASC1_ESRC
ASC1 Error Interrupt Service Request Control Register
ASC1_TBSRC
ASC1 Transmit Buffer Interrupt Service Request Control Register
ASC2_TSRC
ASC2 Transmit Interrupt Service Request Control Register
ASC2_RSRC
ASC2 Receive Interrupt Service Request Control Register
ASC2_ESRC
ASC2 Error Interrupt Service Request Control Register
ASC2_TBSRC
ASC2 Transmit Buffer Interrupt Service Request Control Register

Reset Values: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETR	CLRR	SRR	SRE	0	TOS	0		SRPN							
w	w	rh	rw	r	rw	r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	Service Request Priority Number
TOS	10	rw	Type of Service Control
SRE	12	rw	Service Request Enable
SRR	13	rh	Service Request Flag

Field	Bits	Type	Description
CLRR	14	w	**Request Clear Bit**
SETR	15	w	**Request Set Bit**
0	[9:8], 11, [31:16]	r	**Reserved;** read as 0; should be written with 0.

23.3.3 DMA Requests

The DMA request output lines of the ASC0/ASC1/ASC2 modules become active whenever its related interrupt line is activated. The DMA request lines are connected to the DMA controller as shown in **Table 23-12**.

Table 23-12 DMA Request Lines of ASC0/ASC1

Module	Related ASC Interrupt	DMA Request Line	Description
ASC0	RIR	ASC0_RDR	ASC0 Receive DMA Request
	TIR	ASC0_TDR	ASC0 Transmit DMA Request
ASC1	RIR	ASC1_RDR	ASC1 Receive DMA Request
	TIR	ASC1_TDR	ASC1 Transmit DMA Request
ASC2	RIR	ASC2_RDR	ASC2 Receive DMA Request
	TIR	ASC2_TDR	ASC2 Transmit DMA Request

23.3.4 ASC0/ASC1/ASC2 Register Address Ranges

In the TC1130, the registers of the three ASC modules are located in the following address ranges:

- ASC0 module: Module Base Address = F010 0300_H
  Module End Address = F010 $03FF_H$
- ASC1 module: Module Base Address = F010 0400_H
  Module End Address = F010 $04FF_H$
- ASC2 module: Module Base Address = F010 0500_H
  Module End Address = F010 $05FF_H$
- Absolute Register Address = Module Base Address + Offset Address
  (offset addresses see **Table 23-8**)

Note: The complete and detailed address map of the ASC0/ASC1/ASC2 modules is described in the chapter “Register Overview” of the TC1130 System Units User’s Manual.

24 Synchronous Serial Interface (SSC)

This chapter describes the two SSC synchronous serial interfaces SSC0 and SSC1 of the TC1130. It contains the following sections:

- Functional description of the SSC Kernel, valid for SSC0 and SSC1 (see **Section 24.1**)
- Register descriptions of all SSC Kernel specific registers (see **Section 24.2**)
- TC1130 implementation specific details and registers of the SSC0/SSC1 modules (port connections and control, interrupt control, address decoding, clock control, see **Section 24.3**)

*Note: The SSC kernel register names described in **Section 24.2** will be referenced in the TC1130 User's Manual by the module name prefix "SSC0_" for the SSC0 interface and by "SSC1_" for the SSC1 interface.*

24.1 SSC Kernel Description

Figure 24-1 shows the functional blocks of the SSC interface.

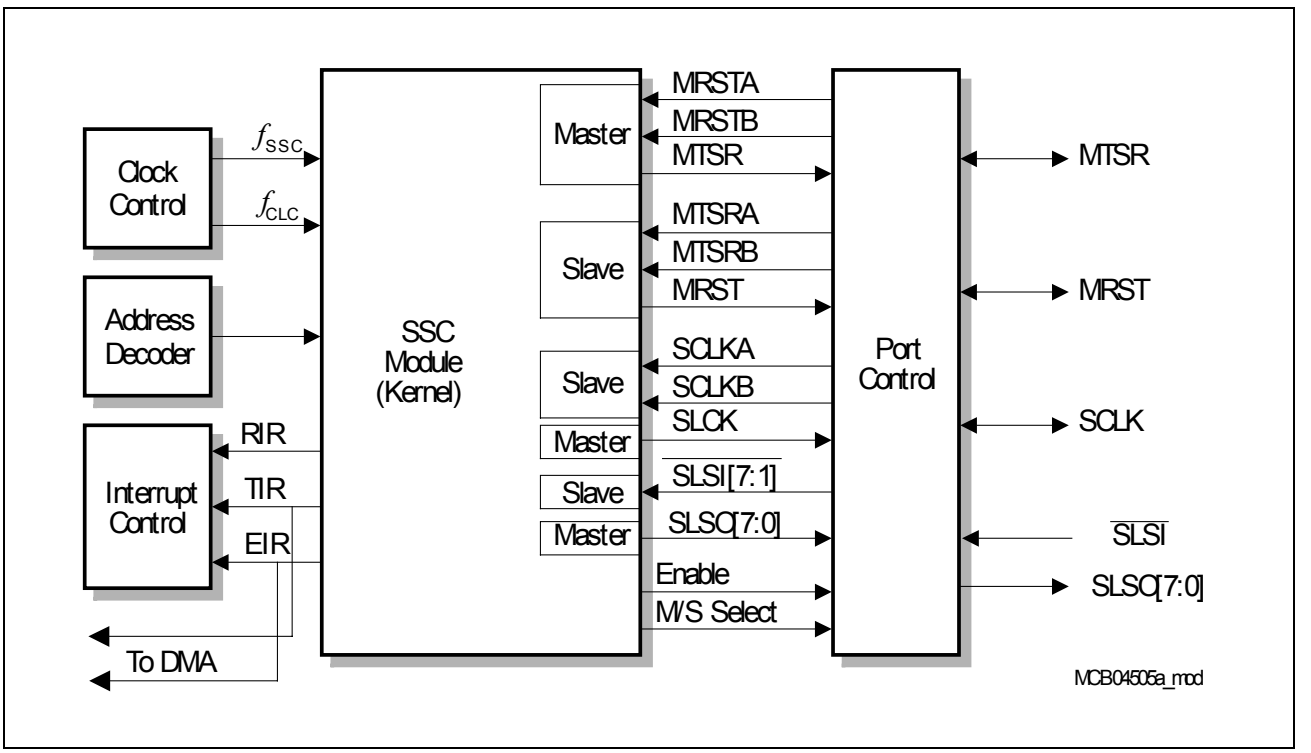


Figure 24-1 General Block Diagram of the SSC Interface

24.1.1 Overview

The SSC supports full-duplex and half-duplex serial synchronous communication up to 37.5 MBaud (@ 75 MHz module clock) with receive and transmit FIFO support. The serial clock signal can be generated by the SSC itself (master mode) or can be received from an external master (slave mode). Data width, shift direction, clock polarity and phase are programmable. This allows communication with SPI-compatible devices. Transmission and reception of data is double-buffered. A shift clock generator provides the SSC with a separate serial clock signal. Eight slave select inputs are available for slave mode operation. Eight programmable slave select outputs (chip selects) are supported in master mode.

Features

- Master and slave mode operation
  - Full-duplex or half-duplex operation
  - Automatic pad control possible
- Flexible data format
  - Programmable number of data bits: 2 to 16 bit
  - Programmable shift direction: LSB or MSB shift first
  - Programmable clock polarity: idle low or high state for the shift clock
  - Programmable clock/data phase: data shift with leading or trailing edge of the shift clock
- Baud rate generation from 37.5 MBaud to 572.2 Baud (@ 75 MHz module clock)
- Interrupt generation
  - On a transmitter empty condition
  - On a receiver full condition
  - On an error condition (receive, phase, baud rate, transmit error)
- Flexible SSC pin configuration
- One slave select inputs $\overline{\text{SLSI}}$ in slave mode
- Eight programmable slave select outputs SLSO in master mode
  - Automatic SLSO generation with programmable timing
  - Programmable active level and enable control
- 4-stage receive FIFO (RXFIFO) and 4-stage transmit FIFO (TXFIFO)
  - Independent control of RXFIFO and TXFIFO
  - 2 to 16 bit FIFO data width
  - Programmable receive/transmit interrupt trigger level
  - Receive and transmit FIFO filling level indication
  - Overrun error generation
  - Underflow error generation

24.1.2 General Operation

The SSC supports full-duplex and half-duplex synchronous communication up to 37.5 MBaud (@ 75 MHz module clock). The serial clock signal can be generated by the SSC itself (master mode) or be received from an external master (slave mode). Data width, shift direction, clock polarity and phase are programmable. This allows communication with SPI-compatible devices. Transmission and reception of data are double-buffered. A shift clock generator provides the SSC with a separate serial clock signal.

Configuration of the high-speed synchronous serial interface is very flexible, so it can work with other synchronous serial interfaces, can serve for master/slave or multimaster interconnections, or can operate compatibly with the popular SPI interface. It can be used to communicate with shift registers (I/O expansion), peripherals (e.g. EEPROMs etc.), or other controllers (networking). The SSC supports half-duplex and full-duplex communication. Data is transmitted or received on pins MTSR (Master Transmit/Slave Receive) and MRST (Master Receive/Slave Transmit). The clock signal is output or input via pin SCLK (Serial Clock). These three pins are typically alternate output functions of port pins. If they are implemented as dedicated bidirectional pins they can be directly controlled by the SSC. In slave mode, the SSC can be selected from a master via dedicated slave select input lines (SLSI). In master mode, automatic generation of slave select output lines (SLSO) is supported.

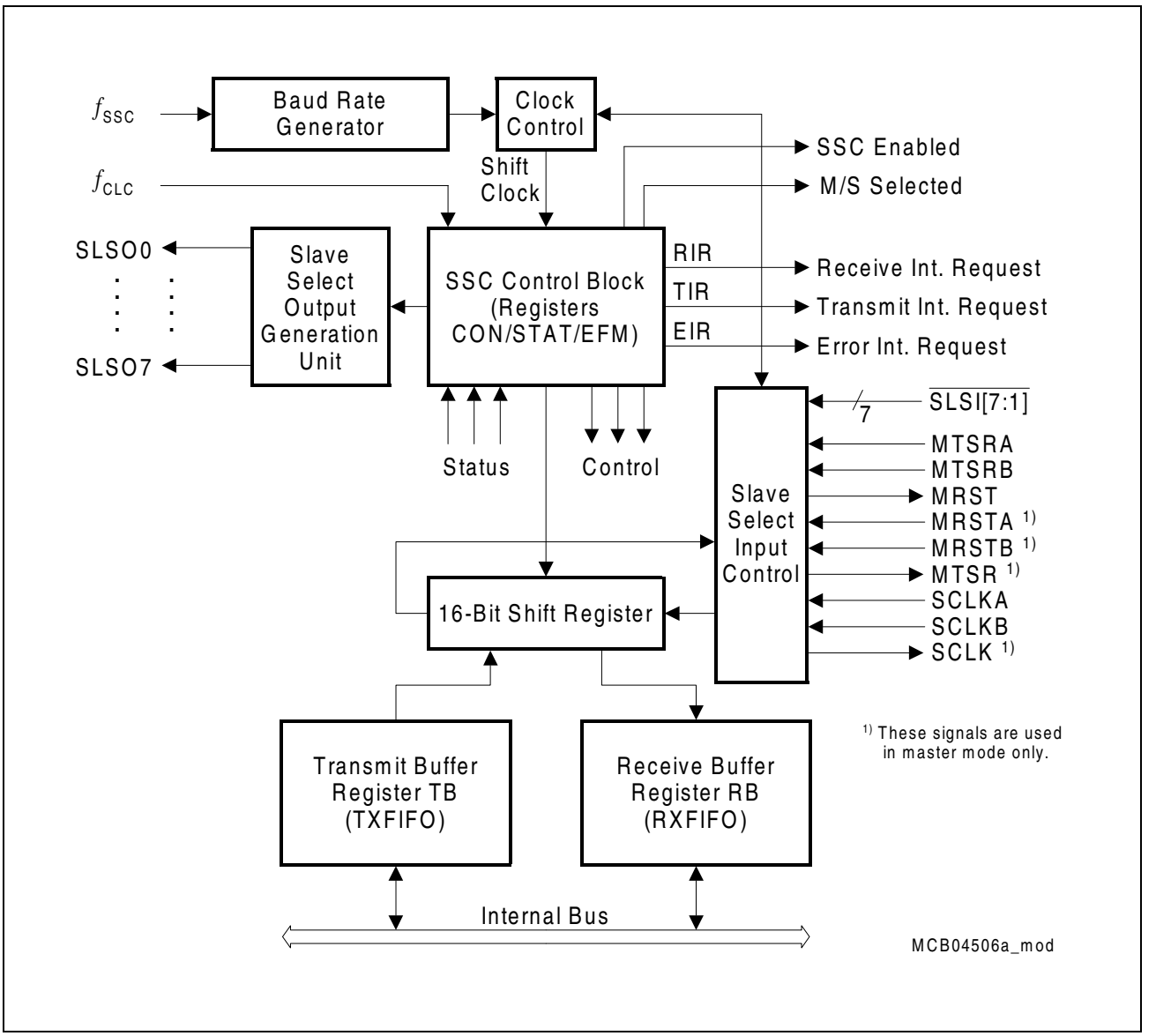


Figure 24-2 Synchronous Serial Channel SSC Block Diagram

24.1.2.1 Operating Mode Selection

The operating mode of the serial channel SSC is controlled by its control register, CON. Status information is contained in its status register, STAT.

The shift register of the SSC is connected to both the transmit pin and the receive pin via the pin control logic (see block diagram in **Figure 24-2**). Transmission and reception of serial data are synchronized and take place at the same time, that is, the same number of transmitted bits is also received. Transmit data is written into the Transmit Buffer TB. It is moved to the shift register as soon as this is empty. An SSC master (CON.MS = 1) immediately begins transmitting, while an SSC slave (CON.MS = 0) will wait for an active shift clock. When the transfer starts, the busy flag STAT.BSY is set and the Transmit Interrupt Request line (TIR) will be activated to indicate that register Transmit Buffer (TB) may be reloaded. When the number of bits (2 to 16, as programmed) have been transferred, the contents of the shift register are moved to the Receive Buffer (RB) and the Receive Interrupt Request line (RIR) will be activated. If no further transfer is to take place (TB is empty), STAT.BSY will be cleared at the same time. Software should not modify STAT.BSY, as this flag is hardware controlled.

Note: Only one SSC (etc.) can be master at a given time.

The transfer of serial data bits can be programmed in many respects:

- The data width can be selected from 2 bits to 16 bits
- A transfer may start with the LSB or the MSB
- The shift clock may be idle low or idle high
- The data bits may be shifted with the leading or trailing edge of the clock signal
- The baud rate (shift clock) can be set from 572.2 Baud up to 37.5 MBaud (@ 75 MHz module clock)
- The shift clock can be generated (master) or received (slave)

These features allow the SSC to be adapted to a wide range of applications that require serial data transfer.

The Data Width Selection supports the transfer of frames of any data length from 2-bit “characters” up to 16-bit “characters”. Starting with the LSB (CON.HB = 0) allows communication with such devices as an SSC device in synchronous mode or 8051-like serial interfaces. Starting with the MSB (CON.HB = 1) allows operation compatible with the SPI interface.
Regardless of the data width selected and whether the MSB or the LSB is transmitted first, the transfer data is always right-aligned in registers TB and RB, with the LSB of the transfer data in bit 0 of these registers. The data bits are rearranged for transfer by the internal shift register logic. The unselected bits of TB are ignored; the unselected bits of RB will not be valid and should be ignored by the receiver service routine.

The Clock Control allows the adaptation of transmit and receive behavior of the SSC to a variety of serial interfaces. A specific clock edge (rising or falling) is used to shift out transmit data, while the other clock edge is used to latch in receive data. Bit CON.PH

selects the leading edge or the trailing edge for each function. Bit CON.PO selects the level of the clock line in the idle state. So for an idle-high clock, the leading edge is a falling one, a 1-to-0 transition (see **Figure 24-3**).

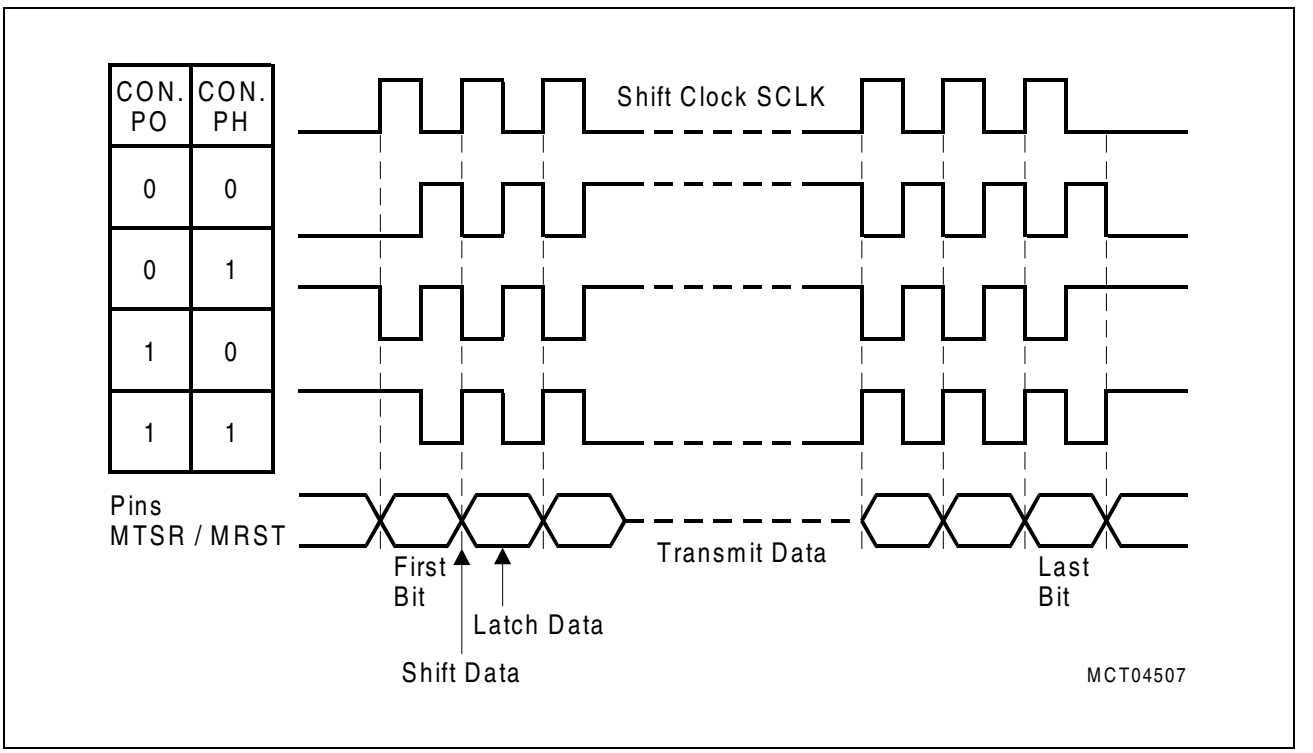


Figure 24-3 Serial Clock Phase and Polarity Options

24.1.2.2 Full-Duplex Operation

*Note: The description in this section assumes that the SSC is used with software controlled bidirectional GPIO port lines that provide open-drain capability (see also **Section 24.1.2.5**).*

The various devices are connected through three lines. The definition of these lines is always determined by the master. The line connected to the master's data output pin MTSR is the transmit line, the receive line is connected to its data input line MRST, and the clock line is connected to pin SCLK. Only the device selected for master operation generates and outputs the serial clock on pin SCLK. All slaves receive this clock, so their pin SCLK must be switched to input mode. The output of the master's shift register is connected to the external transmit line, which in turn is connected to the slaves' shift register input. The output of the slaves' shift register is connected to the external receive line in order to enable the master to receive the data shifted out of the slave. The external connections are hard-wired, with the function and direction of these pins determined by the master or slave operation of the individual device.

*Note: The shift direction shown in **Figure 24-4** applies to both MSB-first and LSB-first operation.*

When initializing the devices in this configuration, one device must be selected for master operation while all other devices must be programmed for slave operation. Initialization includes the operating mode of the device's SSC and also the function of the respective port lines.

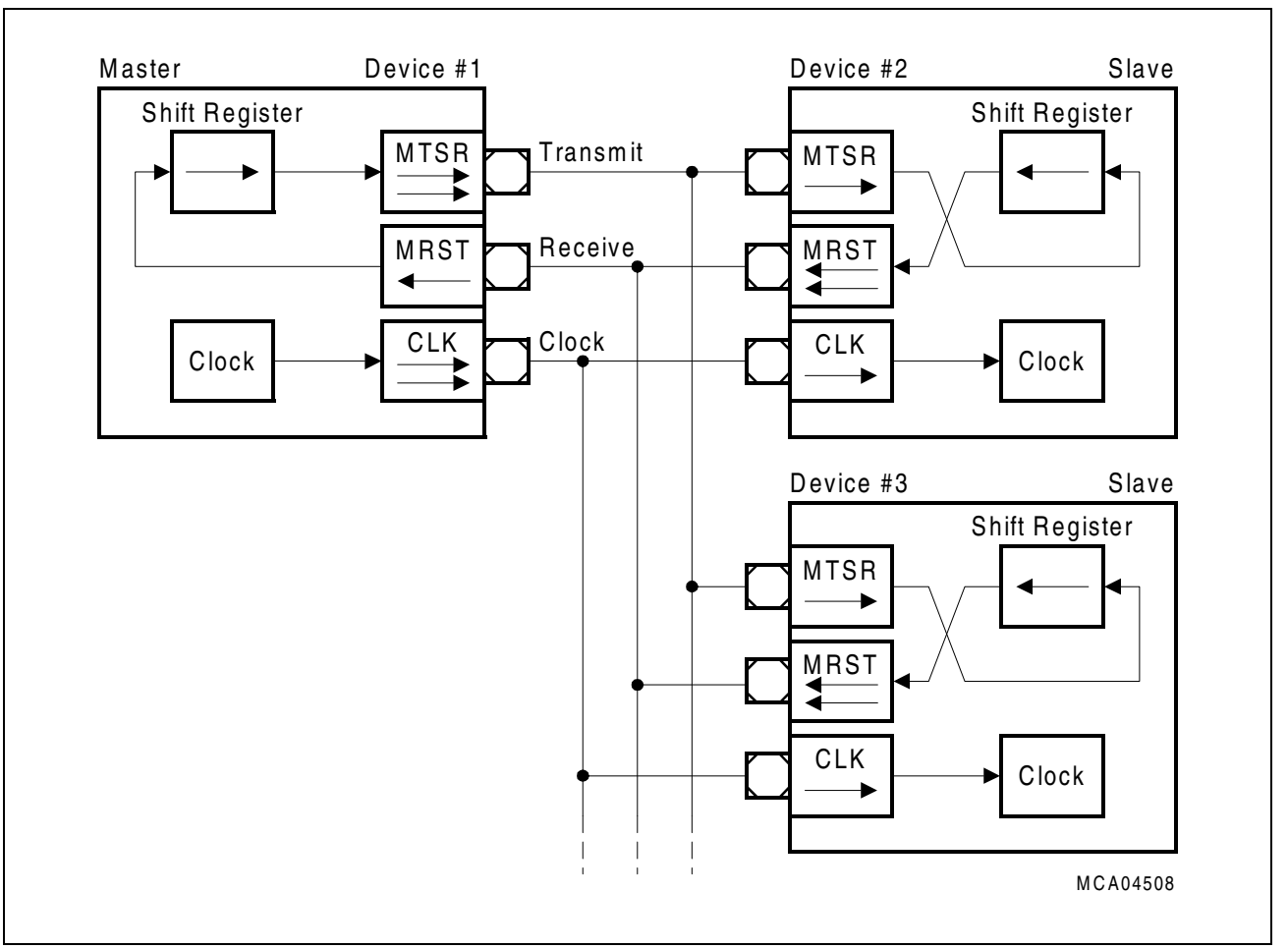


Figure 24-4 SSC Full-Duplex Configuration

The data output pins MRST of all slave devices are connected onto one receive line in this configuration. During a transfer each slave shifts out data from its shift register. There are two ways to avoid collisions on the receive line due to different slave data:

- **Only one slave drives the line** and enables the driver of its MRST pin. All the other slaves must program their MRST pins to input. So, only one slave can put its data onto the master's receive line. Only reception of data from the master is possible. The master selects the slave device from which it expects data either by separate select lines, or by sending a special command to this slave. The selected slave then switches its MRST line to output until it gets a de-selection signal or command.
- **The slaves use open drain output on MRST.** This forms a wired-AND connection. The receive line needs an external pull-up in this case. Corruption of the data on the receive line sent by the selected slave is avoided when all slaves not selected for transmission to the master send only 1s. Since this high level is not actively driven

onto the line, but is only held through the pull-up device, the selected slave can pull this line actively to a low-level when transmitting a zero bit. The master selects the slave device from which it expects data either by separate select lines, or by sending a special command to this slave.

After performing all necessary initializations of the SSC, the serial interfaces can be enabled. For a master device, the alternate clock line will now go to its programmed polarity. The alternate data line will go to either 0 or 1, until the first transfer will start. After a transfer, the alternate data line will always remain at the logic level of the last transmitted data bit.

When the serial interfaces are enabled, the master device can initiate the first data transfer by writing the transmit data into register TB. This value is copied into the shift register (assumed to be empty at this time), and the selected first bit of the transmit data will be placed onto the MTSR line on the next clock from the shift clock generator (transmission only starts, if CON.EN = 1). Depending on the selected clock phase, also a clock pulse will be generated on the SCLK line. With the opposite clock edge, the master simultaneously latches and shifts in the data detected at its input line MRST. This “exchanges” the transmit data with the receive data. Because the clock line is connected to all slaves, their shift registers will be shifted synchronously with the master’s shift register, shifting out the data contained in the registers, and shifting in the data detected at the input line. After the pre-programmed number of clock pulses (via the data width selection), the data transmitted by the master is contained in all slaves’ shift registers, while the master’s shift register holds the data of the selected slave. In the master and all slaves, the content of the shift register is copied into the Receive Buffer (RB) and the Receive Interrupt Line (RIR) is activated.

A slave device will immediately output the selected first bit (MSB or LSB of the transfer data) at pin MRST when the contents of the transmit buffer are copied into the slave’s shift register. Bit STAT.BSY is not set until the first clock edge at SCLK appears. The slave device will not wait for the next clock from the shift clock generator – as the master does – because the first clock edge generated by the master may be already used to clock in the first data bit, depending on the selected clock phase. So the slave’s first data bit must already be valid at this time.

*Note: On the SSC a transmission **and** a reception always takes place at the same time, regardless whether valid data has been transmitted or received.*

24.1.2.3 Half-Duplex Operation

*Note: The description in this section assumes that the SSC is used with software controlled bidirectional GPIO port lines that provide open-drain capability (see also **Section 24.1.2.5**).*

In a half-duplex configuration, only one data line is necessary for both receiving **and** transmitting data. The data exchange line is connected to both pins MTSR and MRST of each device, the clock line is connected to the SCLK pin.

The master device controls the data transfer by generating the shift clock, while the slave devices receive it. Due to the fact that all transmit and receive pins are connected to the one data exchange line, serial data may be moved between arbitrary stations.

Similar to full-duplex mode there are two ways to avoid collisions on the data exchange line:

- Only the transmitting device may enable its transmit pin driver
- The non-transmitting devices use open drain output and only send 1s

Because the data inputs and outputs are connected together, a transmitting device will clock in its own data at the input pin (MRST for a master device, MTSR for a slave). In this way, any corruption is detected on the common data exchange line where the received data is not equal to the transmitted data.

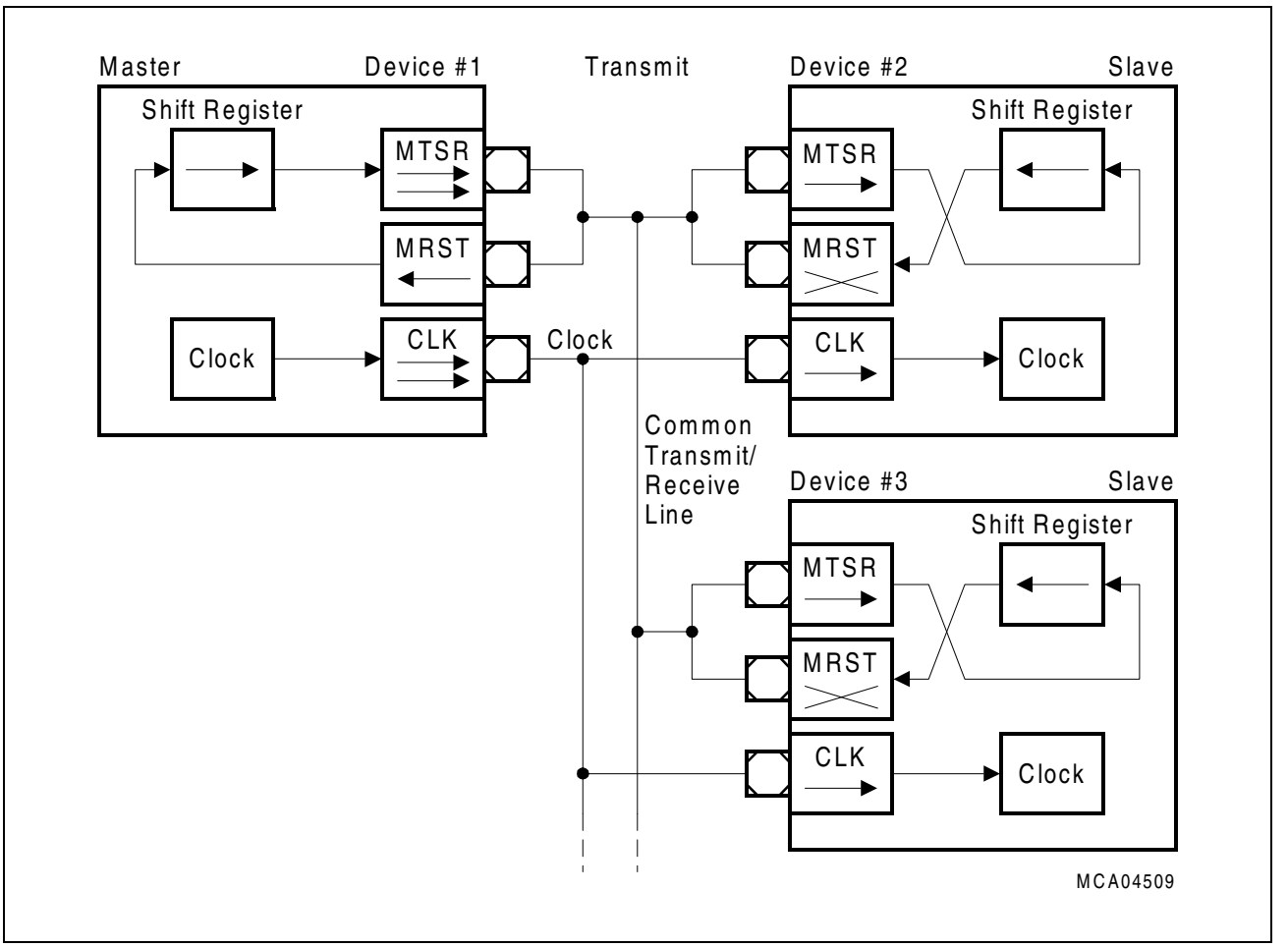


Figure 24-5 SSC Half-Duplex Configuration

24.1.2.4 Continuous Transfers

When the transmit interrupt request flag is set, it indicates that the Transmit Buffer (TB) is empty and is ready to be loaded with the next transmit data. If TB has been reloaded by the time the current transmission is finished, the data is immediately transferred to the shift register and the next transmission can start without any additional delay (according to the selected SLSO timings). On the data line there is no gap between the two successive frames if no delays are selected. For example, two byte transfers would look the same as one word transfer. This feature can be used to interface with devices that can operate with or require more than 16 data bits per transfer. It is just a matter for software how long a total data frame length can be. This option can also be used e.g. to interface to byte-wide and word-wide devices on the same serial bus.

Note: This can only happen in multiples of the selected basic data width, because it would require disabling/enabling of the SSC to reprogram the basic data width on-the-fly.

24.1.2.5 Port Control

The SSC uses three lines to communicate with the external world. Pin SCLK serves as the clock line, while pins MRST (Master Receive/Slave Transmit) and MTSR (Master Transmit/Slave Receive) serve as the serial data input/output lines. As shown in **Figure 24-1** these three lines (SCLK as input, Master Receive, Slave Receive) have all two inputs at the SSC Module kernel. Three bits in register PISEL define which of the two kernel inputs (A or B) are connected. This feature allows for each of the three SSC communication lines to be connected to two inputs coming from different port pins.

Operation of the SSC I/O lines depends on the selected operating mode (master or slave). The direction of the port lines depends on the operating mode. The SSC will automatically use the correct kernel output or kernel input line of the ports when switching modes. Port pins assigned as SSC I/O lines can be controlled in two ways:

- by hardware
- by software

When the SSC I/O lines are connected with dedicated pins typically hardware I/O control should be used. In this case, two output signals reflect directly the state of the CON.EN and CON.MS bits (the M/S select line is inverted to the CON.MS bit definition).

When the SSC I/O lines are connected with bidirectional lines of general purpose I/O ports typically software I/O control should be used. In this case port registers must be programmed for alternate output and input selection. When switching between master and slave mode port registers must be reprogrammed.

Using the open-drain output feature of port lines helps avoid bus contention problems and reduces the need for hard-wired hand-shaking or slave select lines. In this case, it is not always necessary to switch the direction of a port pin. Note that in hardware controlled I/O mode the availability of open-drain outputs depends on the type of the used dedicated output pins. The SSC module itself does not provide any control capability for open drain control.

*Note: Details on SSC port connections and configuration see **Section 24.3.1**.*

24.1.2.6 Transmit FIFO Operation

The transmit FIFO (TXFIFO) provides the following functionality:

- Enable/disable control
- Programmable filling level for transmit interrupt generation
- Filling level indication
- FIFO clear (flush) operation
- FIFO overflow error generation
- 2 to 16 bit TXFIFO data width

The transmit FIFO is controlled by the TXFCON control register. When bit TXFCON.TXFEN is set, the transmit FIFO is enabled. The interrupt trigger level defined by TXFCON.TXFITL defines the filling level of the TXFIFO at which a transmit interrupt TIR is generated. This interrupt is always generated when the filling level of the transmit FIFO is equal to or less than the value stored in TXFCON.TXFITL.

Bit field TXFFL in the FIFO status register FSTAT indicates the number of entries that are actually written (valid) in the TXFIFO. Therefore, the software can verify, in the interrupt service routine, for instance, how many bytes can be still written into the transmit FIFO via register TB without getting an overrun error.

The transmit FIFO cannot be accessed directly. All data write operations into the TXFIFO are executed by writing into the TB register. The data width of one TXFIFO stage can be from 2 to 16 bits (as programmed in CON.BM).

The example in **Figure 24-6** shows an example of a transmit FIFO operation with a typical data width of 8 bits, representing a byte. In this example, four bytes are transmitted via the transmit output line. The transmit FIFO interrupt trigger level TXFCON.TXFITL is set to 0010_B. The first byte written into the empty TXFIFO via TB is directly transferred into the transmit shift register and is not written into the FIFO. After Byte 1, Bytes 2 to 4 are written into the transmit FIFO.

After the transfer of Byte 2 from the TXFIFO into the transmit shift register of the SSC, 2 bytes remain in the TXFIFO. Therefore, the value of TXFCON.TXFITL is reached and a transmit buffer interrupt will be generated at the beginning and a transmit interrupt at the end of the Byte 2 serial transmission. Finally, after the start of the serial transmission of Byte 4, the TXFIFO is again empty.

If the TXFIFO is full and additional bytes are written into TB, the transmit interrupt will be generated with bit CON.TE set if bit CON.TEN was set. In this case, the data that was last written into the transmit FIFO is overwritten and the transmit FIFO filling level FSTAT.TXFFL is set to maximum.

The TXFIFO can be flushed or cleared by setting bit TXFCON.TXFFLU. After this TXFIFO flush operation, the TXFIFO is empty and the transmit FIFO filling level FSTAT.TXFFL is set to 0000_B. A running serial transmission is not aborted by a receive FIFO flush operation.

Note: The TXFIFO is flushed automatically with a reset operation of the SSC module and if the TXFIFO becomes disabled (resetting bit TXFCON.TXFEN) after it was previously enabled.

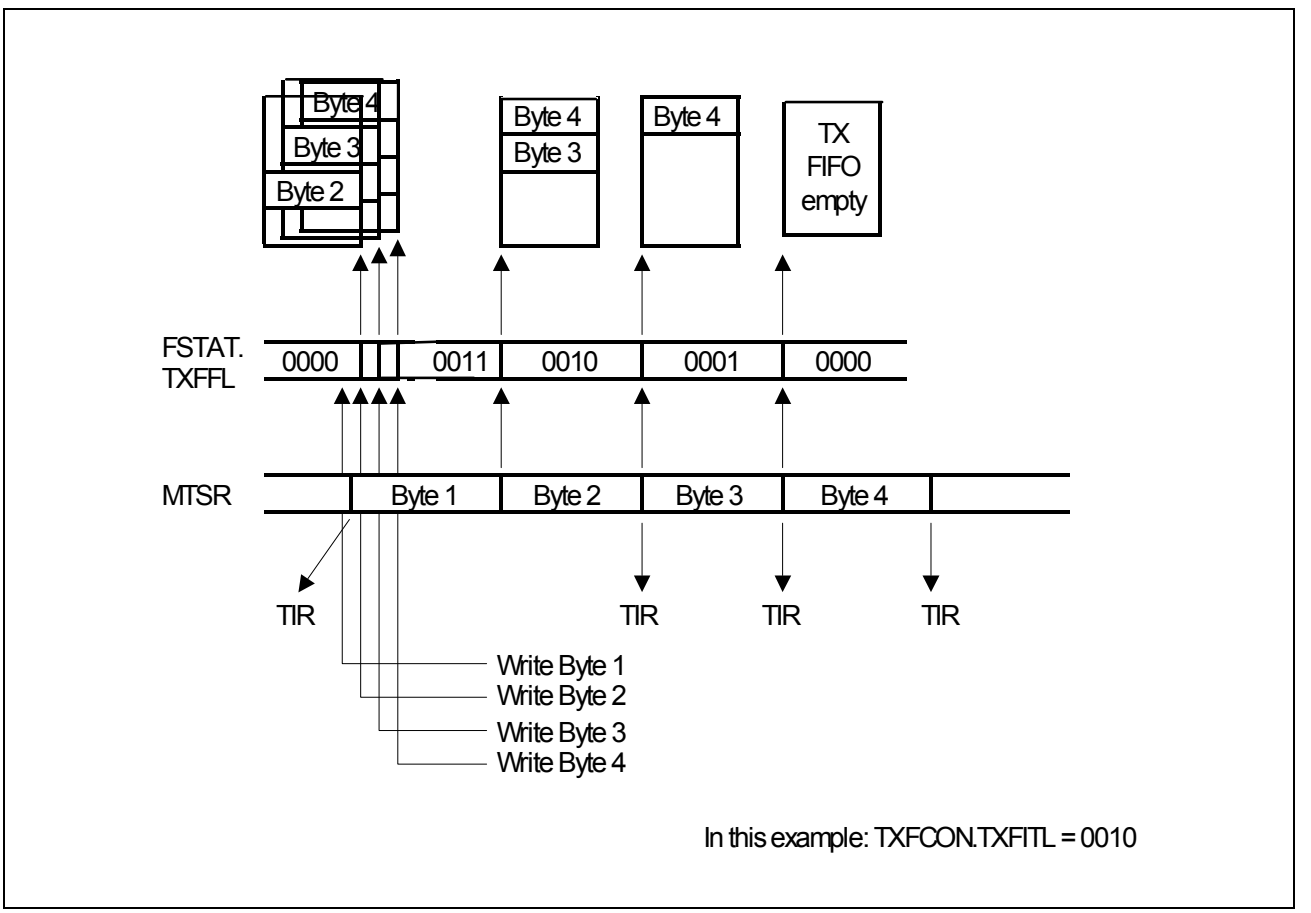


Figure 24-6 Transmit FIFO Operation Example

24.1.2.7 Receive FIFO Operation

The receive FIFO (RXFIFO) provides the following functionality:

- Enable/disable control
- Programmable filling level for receive interrupt generation
- Filling level indication
- FIFO clear (flush) operation
- FIFO overflow error generation
- 2 to 16-bit RXFIFO data width

The receive FIFO is controlled by the RXFCON control register. When bit RXFCON.RXFEN is set, the receive FIFO is enabled. The interrupt trigger level defined by RXFCON.RXFITL defines the filling level of RXFIFO at which a receive interrupt RIR is generated. RIR is always generated when the filling level of the receive FIFO is equal to or greater than the value stored in RXFCON.RXFITL.

Bit field RXFFL in the FIFO status register FSTAT indicates the number of bytes that have been actually written into the FIFO and can be read out of the FIFO by a user program.

The receive FIFO cannot be accessed directly. All data read operations from the RXFIFO are executed by reading the RB register. The data width of one RXFIFO stage can be from 2 to 16 bits (as programmed in CON.BM).

The example in **Figure 24-7** shows an example of a receive FIFO operation with a typical data width of 8 bits, representing a byte. In this example, six bytes are received via the receive input line. The receive FIFO interrupt trigger level RXFCON.RXFITL is set to 0011_B. Therefore, the first receive interrupt RIR is generated after the reception of Byte 3 (RXFIFO is filled with three messages).

After the reception of Byte 4, three bytes are read out of the receive FIFO. After this read operation, the RXFIFO still contains one message. Finally, the FIFO is cleared after reading the last message.

If the RXFIFO is full and additional data are received, the receive interrupt RIR will be generated and bit CON.RE is set, if CON.REN is not cleared. In this case, the data byte last written into the receive FIFO is overwritten. With the overrun condition, the receive FIFO filling level FSTAT.RXFFL is set to maximum. If a RB read operation is executed with the RXFIFO enabled but empty, a receive interrupt RIR will be generated. In this case, the receive FIFO filling level FSTAT.RXFFL is set to 0000_B.

If the RXFIFO is available but disabled (RXFCON.RXFEN = 0) the receive operation is functionally equivalent to the receive operation of the SSC module without FIFO.

The RXFIFO can be flushed or cleared by setting bit RXFCON.RXFFLU in register RXFCON. After this RXFIFO flush operation, the RXFIFO is empty and the receive FIFO filling level FSTAT.RXFFL is set to 0000_B.

The RXFIFO is flushed automatically with a reset operation of the SSC module and if the RXFIFO becomes disabled (resetting bit RXFCON.RXFEN) after it was previously

enabled. Resetting bit CON.REN without resetting RXFCON.RXFEN does not affect (reset) the RXFIFO state. This means that the receive operation of the SSC is stopped, in this case, without changing the content of the RXFIFO. After setting CON.REN again, the RXFIFO with its content is again available.

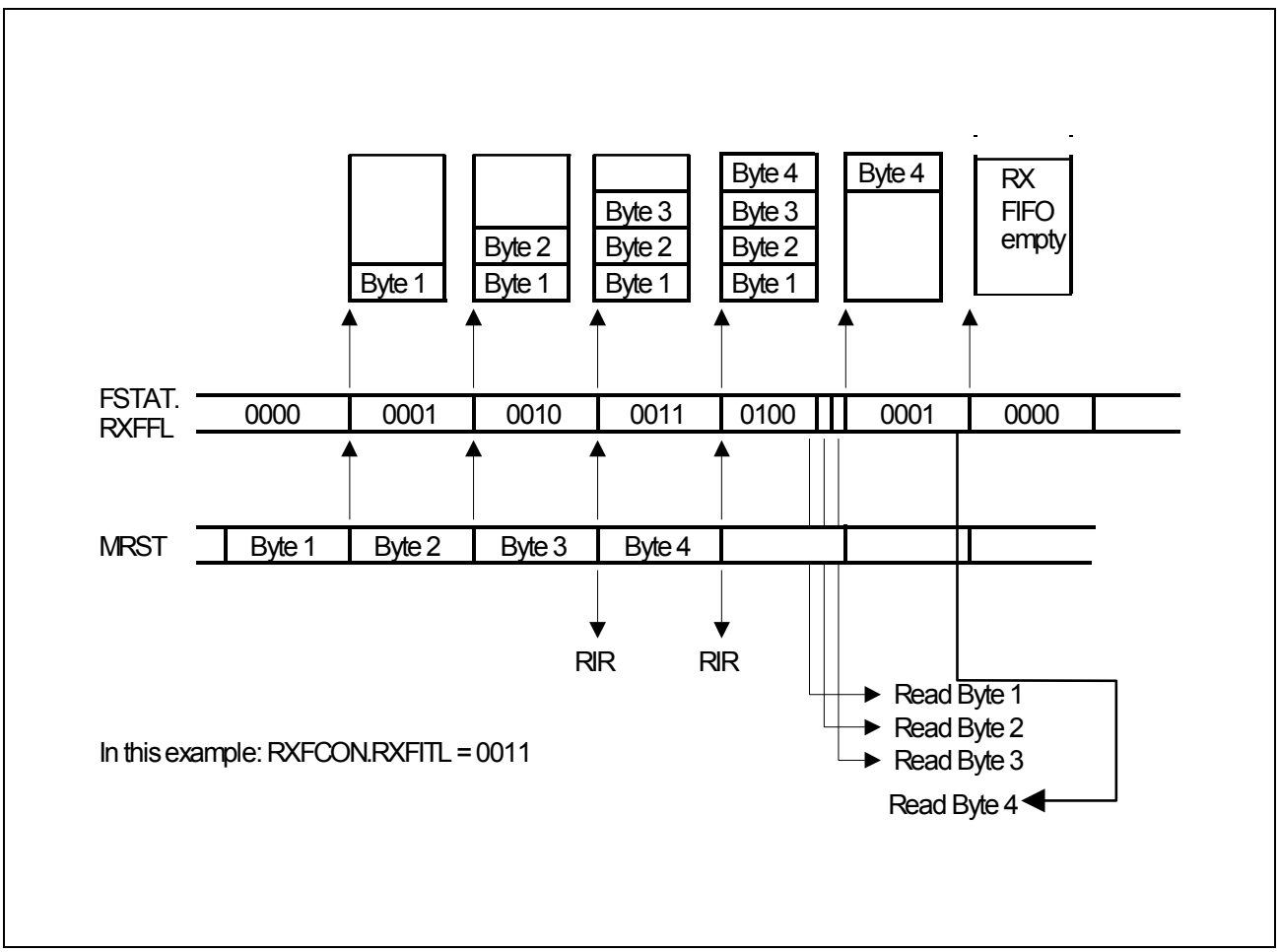


Figure 24-7 Receive FIFO Operation Example

24.1.2.8 FIFO Transparent Mode

In Transparent Mode, a specific interrupt generation mechanism is used for receive and transmit interrupts. In Transparent Mode, receive interrupts are always generated if data bytes are available in the RXFIFO. The relevant conditions for interrupt generation in Transparent Mode are:

- FIFO filling level
- Read/Write operations from/to the RB/TB data registers

Receive Operation

The interrupt generation for the receive FIFO depends on the RXFIFO filling level and the execution of read operations of register RB (see **Figure 24-8**). Transparent Mode for the RXFIFO is enabled when bits RXFCON.RXTMEN and RXFCON.RXFEN in register RXFCON are set.

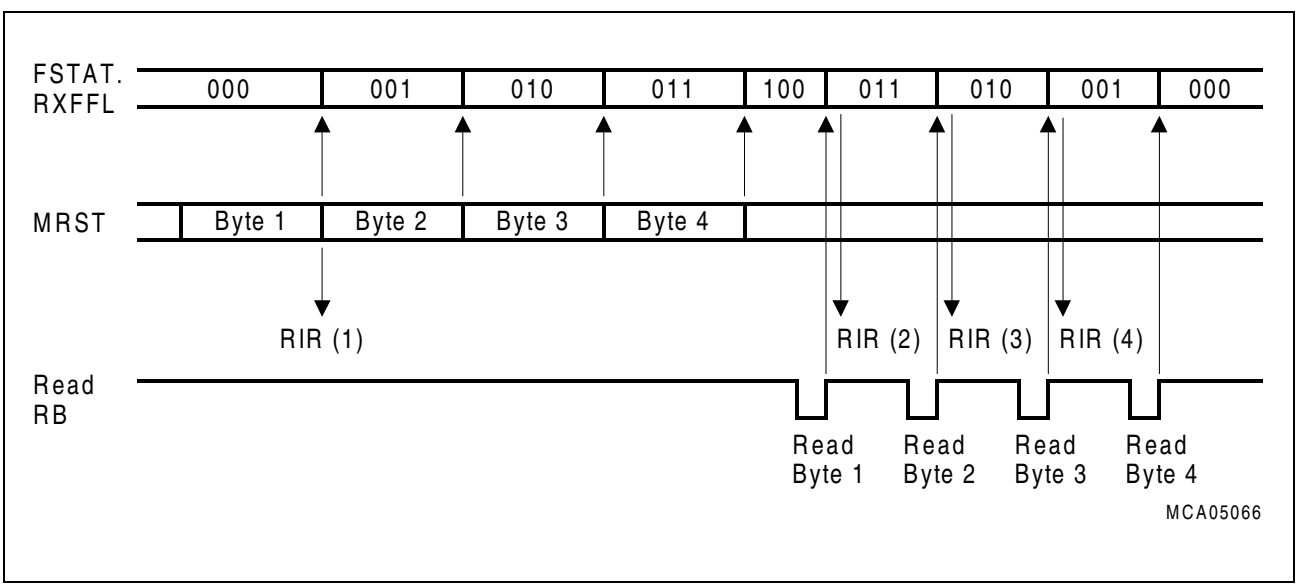


Figure 24-8 Transparent Mode Receive FIFO Operation

If the RXFIFO is empty, a receive interrupt RIR is always generated when the first message is written into an empty RXFIFO (FSTAT.RXFFL changes from 0000_B to 0001_B). If the RXFIFO is filled with at least one message, the occurrence of further receive interrupts depends on the read operations of register RB. The receive interrupt RIR will always be activated after a RB read operation if the RXFIFO still contains data (FSTAT.RXFFL is not equal to 0000_B). If the RXFIFO is empty after a RB read operation, no further receive interrupt will be generated.

If the RXFIFO is full (FSTAT.RXFFL = 1000_B) and additional messages are received, a receive interrupt RIR will be generated. In this case, the message last written into the receive FIFO is overwritten. If a RB read operation is executed with the RXFIFO enabled but empty (underflow condition), a receive interrupt RIR will be generated as well, with bit CON.RE set.

If the RXFIFO is flushed in Transparent Mode, the software must take care that a previous pending receive interrupt is ignored.

Note: The Receive FIFO Interrupt Trigger Level bit field RXFCON.RXFITL is not applicable in Transparent Mode.

Transmit Operation

Interrupt generation for the transmit FIFO depends on the TXFIFO filling level and the execution of write operations to the register TB. Transparent Mode for the TXFIFO is enabled when bits TXFCON.TXTMEN and TXFCON.TXFEN are set.

TIR is also activated after a TXFIFO flush operation or when the TXFIFO becomes enabled (TXFCON.TXTMEN and TXFCON.TXFEN set) when it was previously disabled. In these cases, the TXFIFO is empty and ready to be filled with data.

If the TXFIFO is full (FSTAT.TXFFL = 1000_B) and an additional message is written into TB, a transmit interrupt will be generated after the TB write operation. In this case the data byte last written into the transmit FIFO is overwritten and a transmit interrupt (TIR) will be generated with bit CON.TE set.

Note: The Transmit FIFO Interrupt Trigger Level bit field TXFCON.TXFITL is not applicable in Transparent Mode.

24.1.2.9 Baud Rate Generation

The serial channel SSC has its own dedicated 16-bit baud-rate generator with 16-bit reload capability, allowing baud rate generation independent from the timers. In addition to **Figure 24-2**, **Figure 24-9** shows the baud-rate generator of the SSC in more detail.

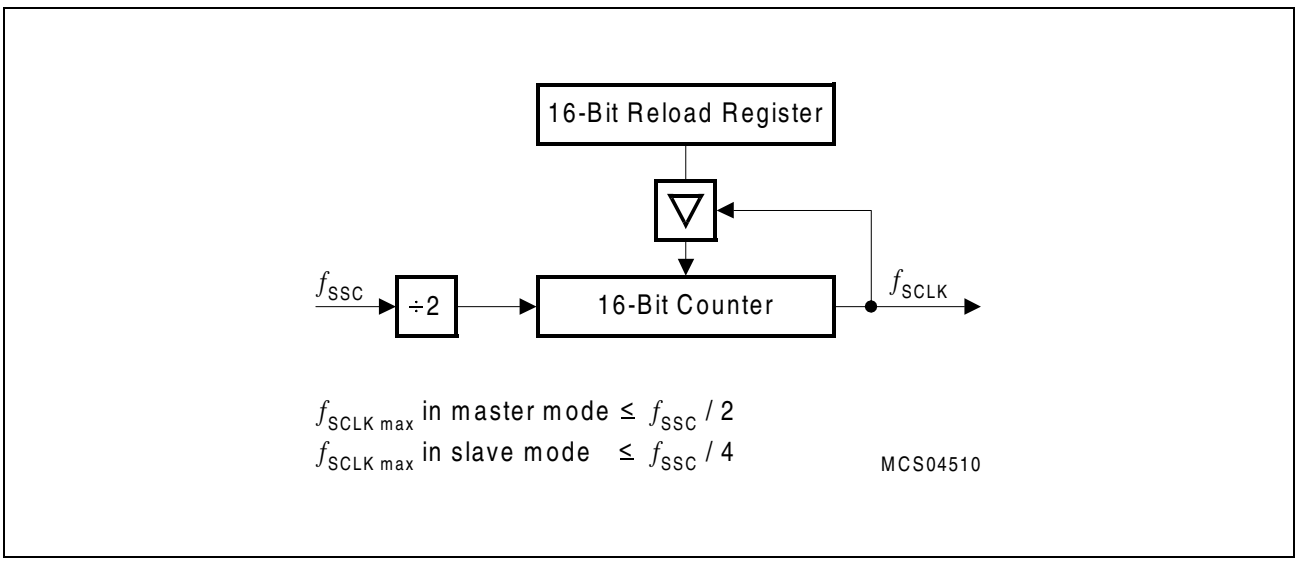


Figure 24-9 SSC Baud-Rate Generator

The baud-rate generator is clocked with the module clock f_{SSC}. The timer counts downwards. Register BR is the dual-function Baud-rate Generator/Reload register. Reading BR, while the SSC is enabled, returns the contents of the timer. Reading BR, while the SSC is disabled, returns the programmed reload value. In this mode, the desired reload value can be written to BR.

Note: Never write to BR while the SSC is enabled.

The formulas below calculate either the resulting baud rate for a given reload value, or the required reload value for a given baud rate:

$$\text{Baudrate}_{SSC} = \frac{f_{SSC}}{2 \times (\text{BR\_VALUE} + 1)} \qquad \text{BR\_VALUE} = \frac{f_{SSC}}{2 \times \text{Baudrate}_{SSC}} - 1 \qquad (24.1)$$

BR_VALUE represents the content of the reload register, taken as unsigned 16-bit integer while Baud rate$_{SSC}$ is equal to f_{SCLK} as shown in **Figure 24-9**.

The maximum baud rate that can be achieved when using a module clock of 75 MHz is 37.5 MBaud in master mode (with
 = 0000_H) and 18.75 MBaud in slave mode (with
 = 0001_H).

Table 24-1 lists some possible baud rates together with the required reload values and the resulting bit times, assuming a module clock of 75 MHz.

Table 24-1 Typical Baud Rates of the SSC (f_{SSC} = 75 MHz)

Reload Value	Baud Rate (= f_{SCLK})	Deviation
0000_H	37.5 MBaud (only in master mode)	0.0%
0001_H	18.75 MBaud	0.0%
0025_H	1 MBaud	-1.3%
0176_H	100 kBaud	0.0%
$0EA5_H$	10 kBaud	0.0%
9276_H	1 kBaud	0.0%
$FFFF_H$	572.2 Baud	0.0%

24.1.2.10 Slave Select Input Operation

For systems with multiple slaves, the SSC module provides $\overline{\text{SLSI}}$ slave select input lines that allow the enabling/disabling of the SCLK, MTSR, and MRST signals in slave mode. Slave mode is selected by CON.MS = 0. The $\overline{\text{SLSI}}$ input logic shown in **Figure 24-10** is controlled by register PISEL and CON.

With PISEL.SLSIS = 0 and slave mode selected, the $\overline{\text{SLSI}}$[7:1] lines do not control the SSC I/O lines. The slave receive input signal at pins MTSRA or MTSRB and the slave clock signal at pin SCLKA or SCLKB are passed further to MTSRI and SCLKI. The slave transmit signal MRSTI is passed directly to MRST.

With PISEL.SLSIS = 1, slave select mode is enabled and input signals $\overline{\text{SLSI}}$[7:1] control the operation of the SSC I/O lines as follows:

- $\overline{\text{SLSIx}}$ = 1: SSC slave is not selected
  - MTSRI is connected with the slave receive input signals MTSRA or MTSRB, depending on PISEL.SRIS (slave mode receive input select)
  - MRST is driven with the logic level of bit PISEL.STIP (slave transmit idle state)
  - SCLKI is driven with the logic level of CON.PO (clock polarity control)
- $\overline{\text{SLSIx}}$ = 0: SSC is selected as slave
  - MTSRI is connected with the slave receive input signals MTSRA or MTSRB, depending on PISEL.SRIS (slave mode receive input select)
  - MRST is directly driven with the slave transmit output signal MRSTI
  - SCLKI is connected with the slave clock input signals SCLKA or SCLKB, depending on PISEL.SCIS (slave mode clock input select)

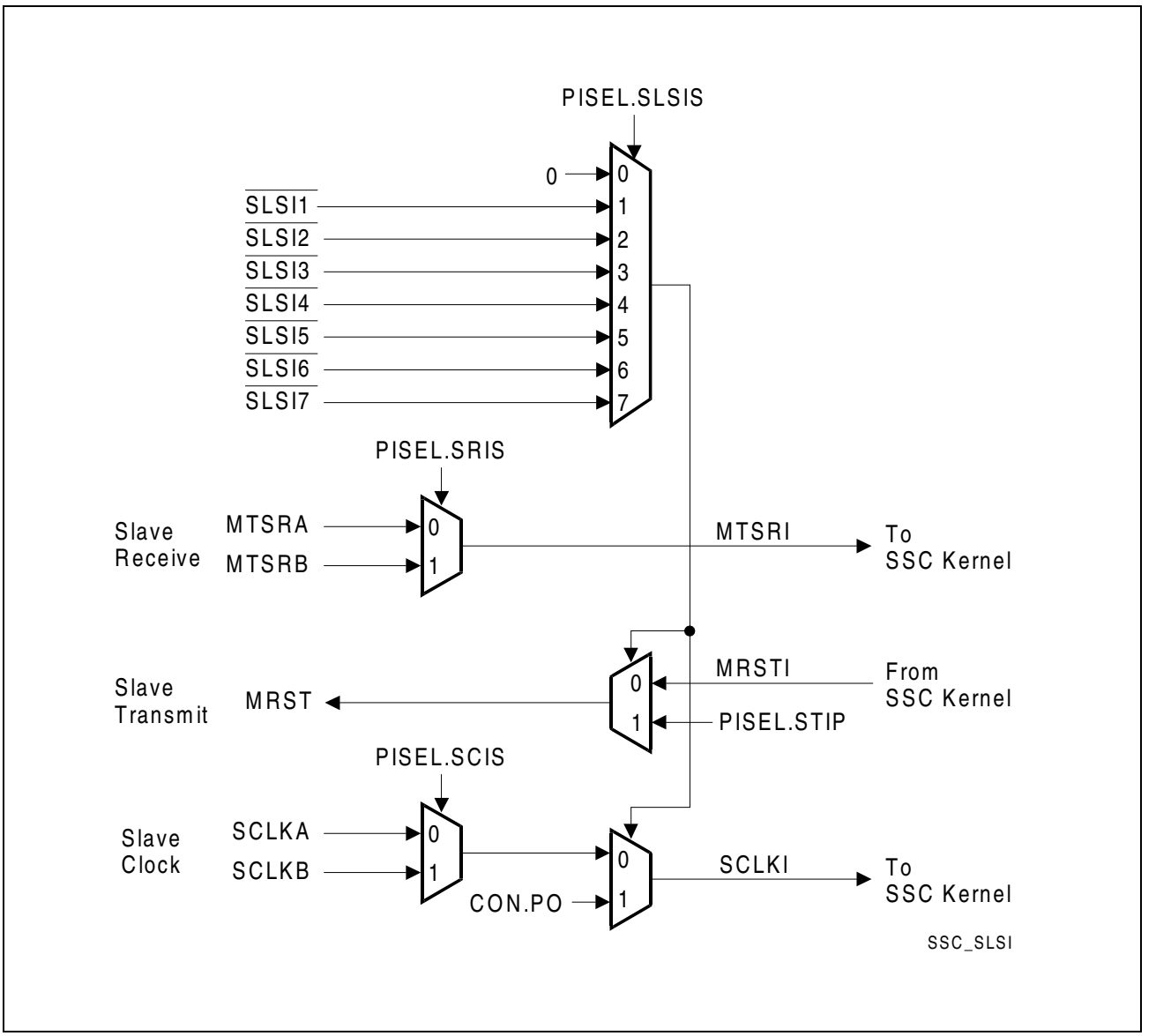


Figure 24-10 Slave Select Input Logic

24.1.2.11 Slave Select Output Generation Unit

In master mode the slave select output generation unit of the SSC provides an automatic generation of up to eight slave select output lines for serial transmit operations. The slave select output generation unit also allows the chip select timing parameters to be adjusted. The active/inactive state of a slave select output as well as the enable/disable state can be controlled individually for each slave select output (see **Figure 24-12**). The basic slave select output timing is shown in **Figure 24-11**, assuming a low active level of the SLSOn lines.

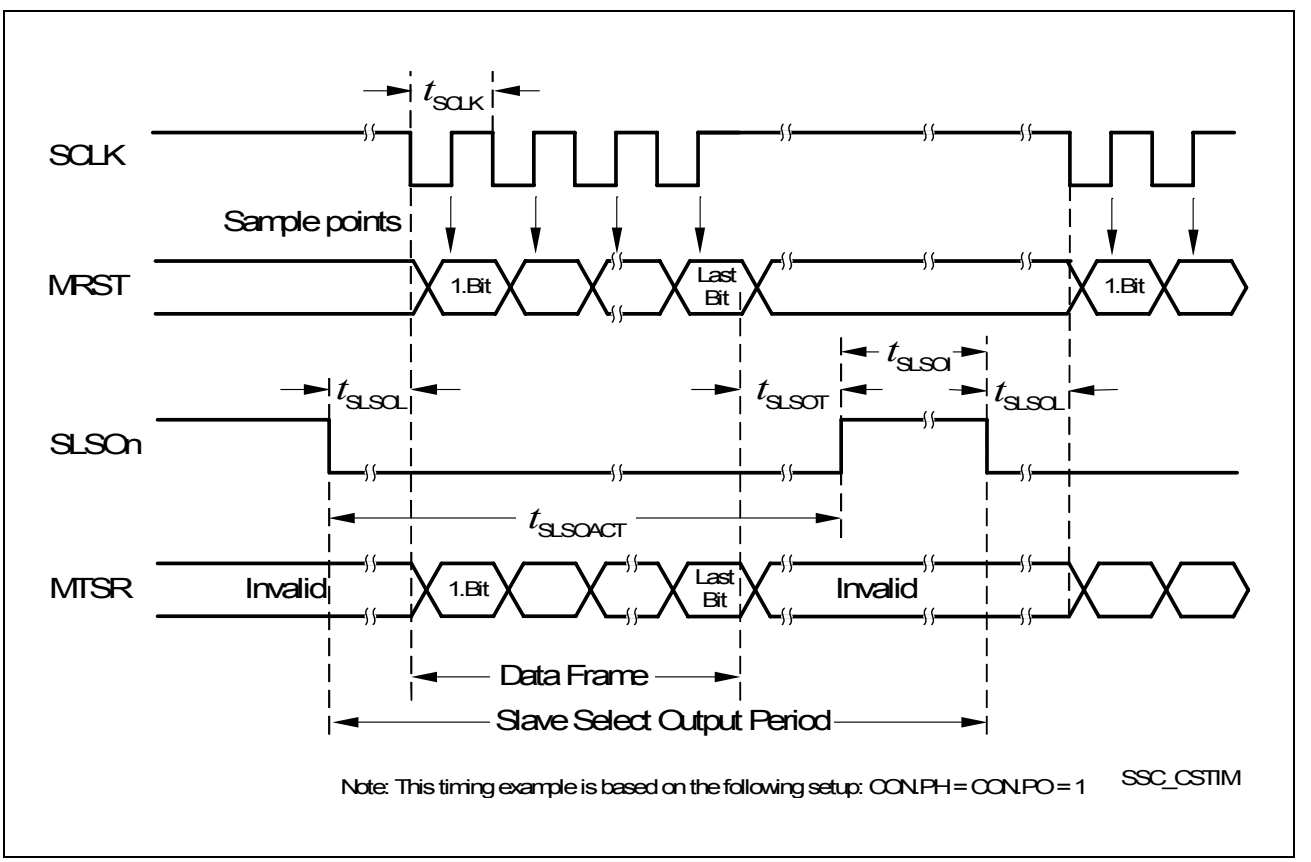


Figure 24-11 SSC Slave Select Output Timing

A slave select output period always starts after a serial write operation to register TB. Afterwards SLSOn becomes active (low) for a number of SCLK cycles (leading delay cycles) before the first bit of the serial data stream occurs at MTSR. After the transmission of the data frame SLSOx remains active (low) for a number of SCLK cycles (trailing delay cycles) before it becomes again inactive. This inactive state of SLSOn is valid at least for a number of SCLK cycles (inactive delay cycles) before a new chip select period can be started.

Note: When operating in master mode with CON.PH = 1 and sampling data from a slave device which becomes enabled by a SLSOx output, a leading delay of at least one leading delay clock cycle should be selected. Because with CON.PH = 1, the first SCLK edge already latches the first data bit at MRST.

The three parameters of a chip select period are controlled by bit fields in the slave select output timing control register SSOTC. Each of these bit fields can contain a value from 0 to 3 defining delay cycles of 0 to 3 multiples of the t_{SCLK} shift clock period. The three parameters are:

- Number of leading delay cycles (t_{SLSOL} = SSOTC.LEAD × t_{SCLK})
- Number of trailing delay cycles (t_{SLSOT} = SSOTC.TRAIL × t_{SCLK})
- Number of inactive delay cycles (t_{SLSOI} = SSSOTC.INACT × t_{SCLK})

If SSOTC.INACT = 00_B and register TB has already been loaded with the data for the next data frame, the next chip select period is started with its leading delay phase without SLSOn going inactive. If, in this case, TB has not been loaded in time with the data for the next data frame, SLSOx becomes inactive again.

Slave Select Output Control

Each slave select output SLSOn can be enabled individually. When SSOC.OENn = 0, SLSOn is enabled. Further, active and inactive levels of the SLSOn outputs are programmable. Bit SSOC.AOLn defines the state of the active level of SLSOn.

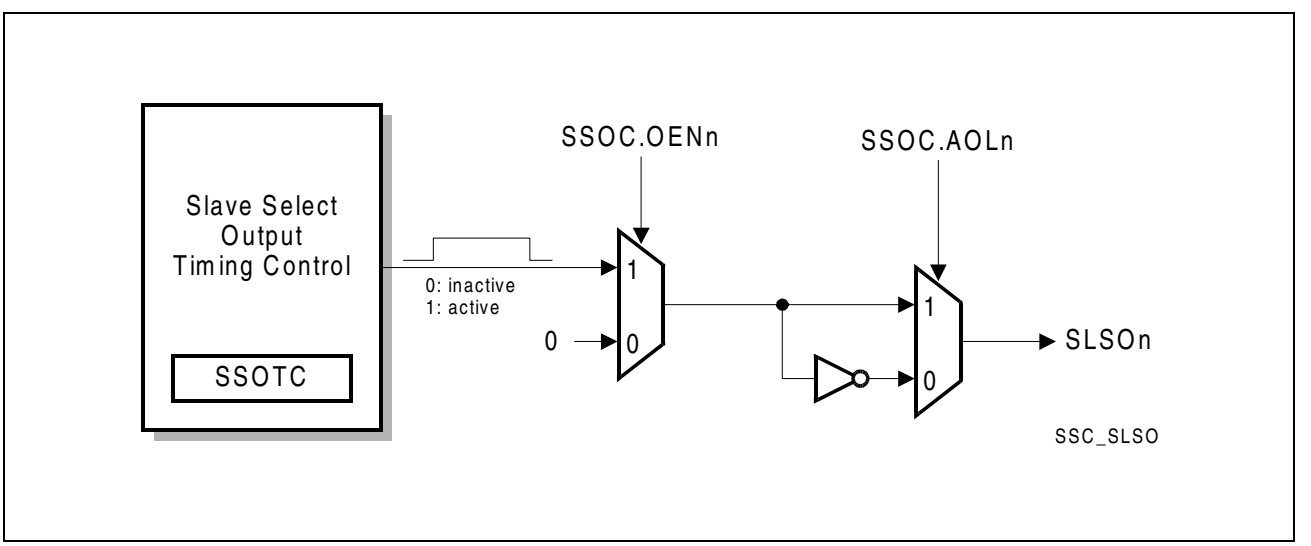


Figure 24-12 Slave Select Output Control Logic

Slave Select Output 7 Delayed Mode

In the SLSO7 delayed mode (SSOTC.SLSO7MOD = 1), the timing of the slave select output SLSO7 as programmed by the three parameters in SSOTC (number of trailing, leading, and inactive delay clock cycles) is delayed by one shift clock period for the inactive to active edge. The active to inactive edge is not delayed. The timing of SLSO7 in the delayed mode is shown in **Figure 24-13**. The bold lines show the timing of SLSO7 in normal operating mode and the dotted lines show the timing of SLSO7 in delayed mode.

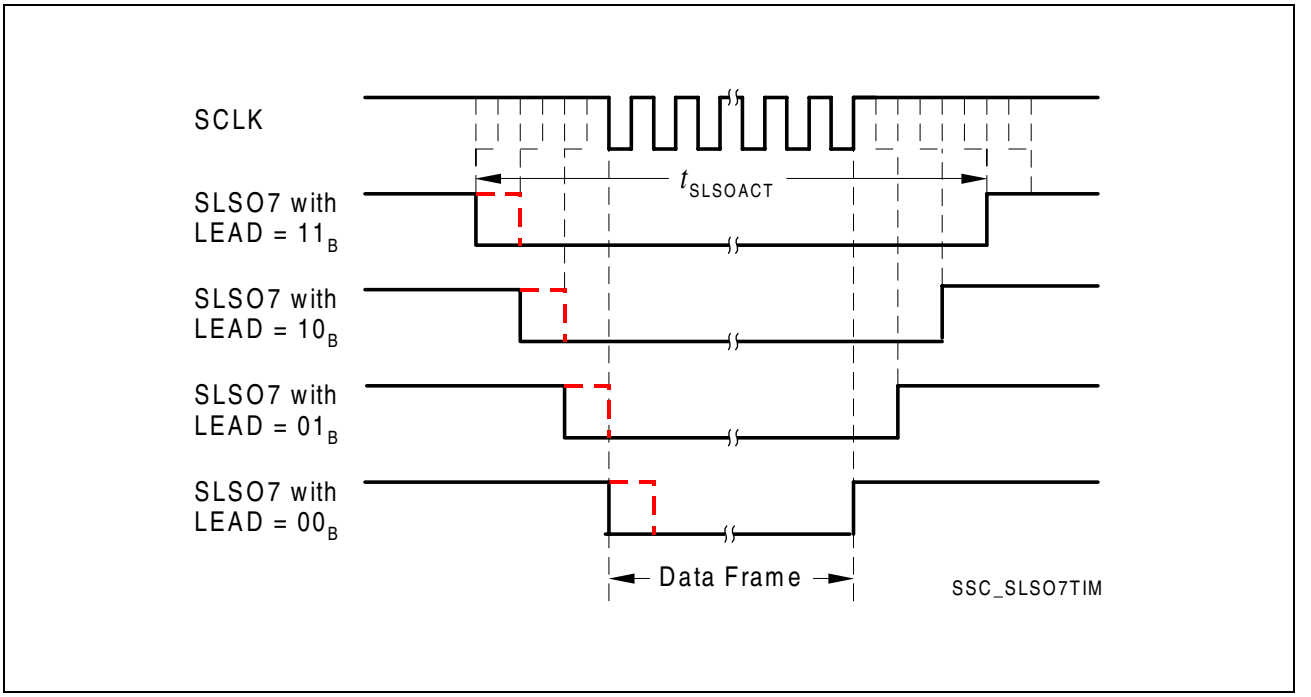


Figure 24-13 SLSO7 Delayed Mode

Slave Select Register Update

The bits in the registers SSOC and SSOTC are buffered while a transfer is in progress. The buffer samples the values written to these registers in the following case:

- Start of the internal transfer sequence

So it is always guaranteed that the data of one SSC transfer is transmitted with one constant slave select configuration and a configuration change is only valid with the start of the next new SSC transfer.

24.1.2.12 Shift Clock Generation

The serial channel SSC operates with its own shift clock f_{SSC}.The shift clock is generated outside the SSC module kernel in the clock control block using a fractional divider (see **Section 24.3.3**).

24.1.2.13 Error Detection Mechanisms

The SSC is able to detect four different error conditions. Receive Error and Phase Error are detected in all modes, while Transmit Error and Baud Rate Error apply to slave mode only. When an error is detected, the respective error flag is set and an error interrupt request will be generated by activating the EIR line (see **Figure 24-14**). The error interrupt handler may then check the error flags to determine the cause of the error interrupt. The error flags are not reset automatically, but must be cleared via register EFM after servicing. This allows servicing of some error conditions via interrupt, while others may be polled by software. The error status flags can be set and reset by software via the error flag modification register EFM.

Note: The error interrupt handler must clear the associated (enabled) error flag(s) to prevent repeated interrupt requests.

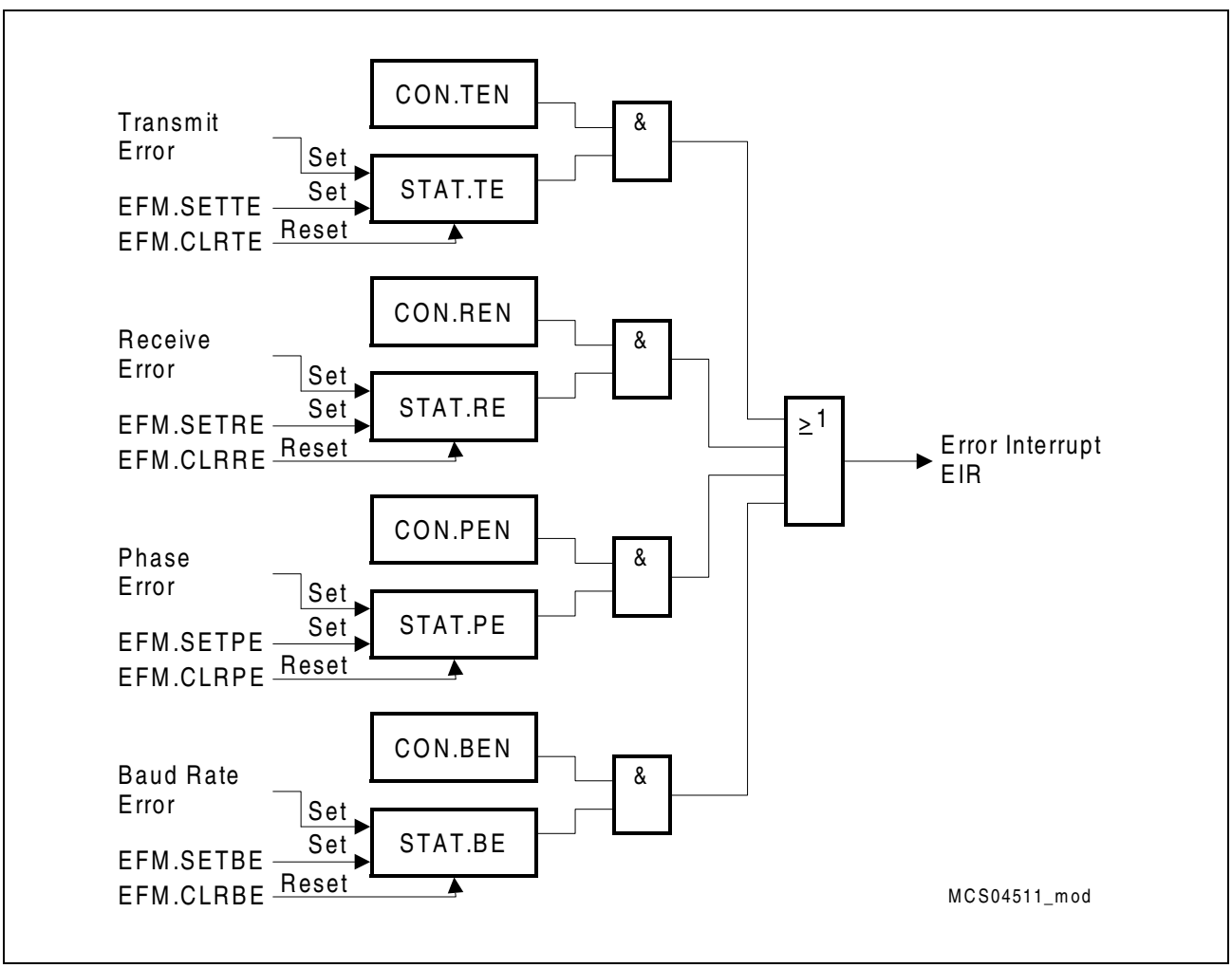


Figure 24-14 SSC Error Interrupt Control

A **Receive Error** (Master or Slave mode) is detected when a new data frame is completely received, but the previous data was not read out of the receive buffer register RB. This condition sets the error flag STAT.RE, when enabled via CON.REN, the error

interrupt request activates the EIR line. The old data in the receive buffer RB will be overwritten with the new value and is unretrievably lost.

A **Phase Error** (Master or Slave mode) is detected when the incoming data at pin MRST (master mode) or MTSR (slave mode), sampled with the same frequency as the module clock, changes between one cycle before and two cycles after the latching edge of the shift clock signal SCLK. This condition sets the error status flag STAT.PE, when enabled via CON.PEN, the error interrupt request activates the EIR line.

A **Baud Rate Error** (Slave mode) is detected when the incoming clock signal deviates from the programmed baud rate (shift clock) by more than 100%, meaning it is either more than double or less than half the expected baud rate. This condition sets the error status flag STAT.BE, when enabled via CON.BEN, the error interrupt request activates the EIR line. Using this error detection capability requires that the slave's shift clock generator is programmed to the same baud rate as the master device. This feature detects false additional pulses or missing pulses on the clock line (within a certain frame).

Note: If this error condition occurs and bit CON.REN = 1, an automatic reset of the SSC will be performed in case of this error. This is done to re-initialize the SSC, if too few or too many clock pulses have been detected.

A **Transmit Error** (Slave mode) is detected when a transfer was initiated by the master (shift clock gets active), but the Transmit Buffer (TB) of the slave was not updated since the last transfer. This condition sets the error status flag STAT.TE, when enabled via CON.TEN, the error interrupt request activates the EIR line. If a transfer starts while the transmit buffer is not updated, the slave will shift out the 'old' contents of the shift register, which is normally the data received during the last transfer. This may lead to the corruption of the data on the transmit/receive line in Half-duplex Mode (open drain configuration) if this slave is not selected for transmission. This mode requires that slaves not selected for transmission only shift out ones, thus, their transmit buffers must be loaded with $FFFF_H$ prior to any transfer.

Note: A slave with push/pull output drivers not selected for transmission, will normally have its output drivers switched. However, to avoid possible conflicts or misinterpretations, it is recommended to always load the slave's transmit buffer prior to any transfer.

The cause of an error interrupt request (receive, phase, baud rate, transmit error) can be identified by the error status flags in control register CON.

Note: In contrast to the error interrupt request line EIR, the error status flags STAT.TE, STAT.RE, STAT.PE, and STAT.BE, are not reset automatically upon entry into the error interrupt service routine, but must be cleared by software.

24.2 SSC Kernel Registers

Figure 24-15 and Table 24-2 show all registers associated with the SSC Kernel.

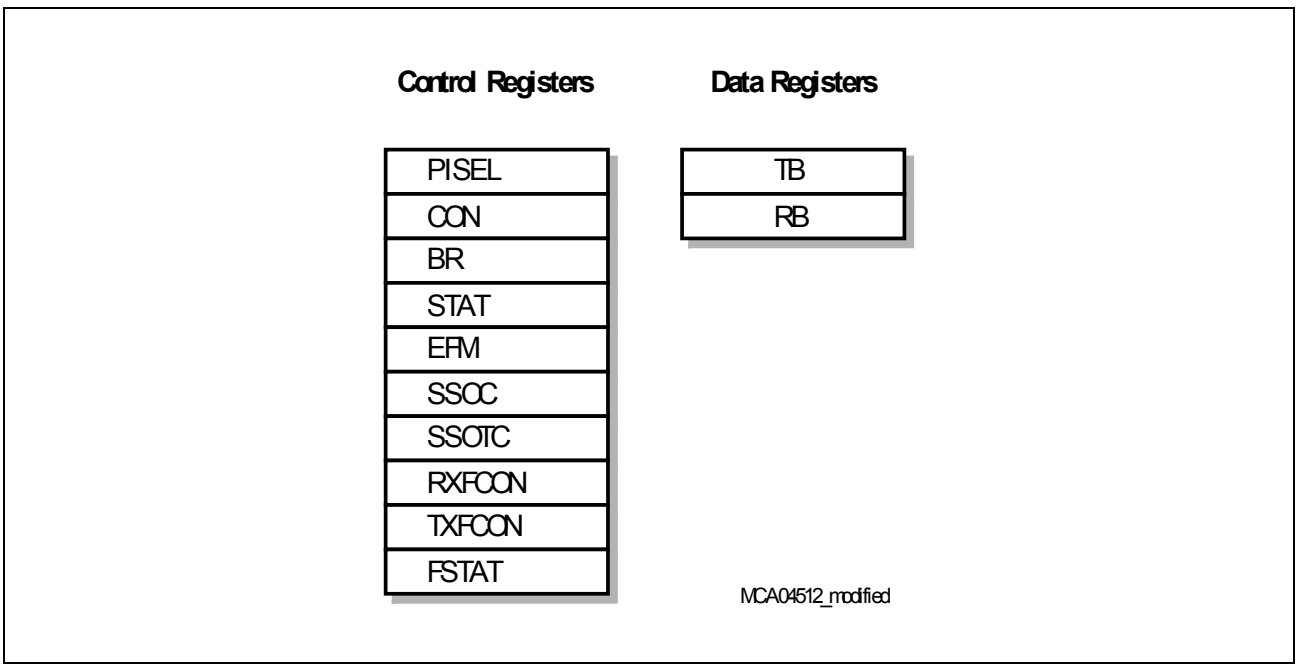


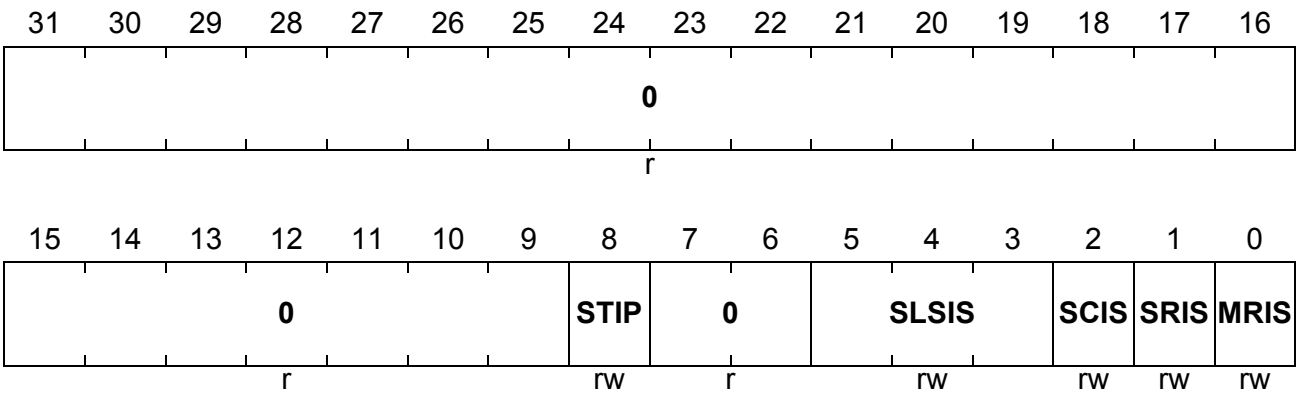
Figure 24-15 SSC Kernel Registers

Table 24-2 SSC Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
PISEL	Port Input Select Register	0004_H	Page 24-29
CON	Control Register	0010_H	Page 24-31
BR	Baud Rate Timer Reload Register	0014_H	Page 24-38
STAT	Status Register	0028_H	Page 24-33
EFM	Error Flag Modification Register	$002C_H$	Page 24-34
SSOC	Slave Select Output Control Register	0018_H	Page 24-36
SSOTC	Slave Select Output Timing Control Register	$001C_H$	Page 24-37
TB	Transmit Buffer Register	0020_H	Page 24-39
RB	Receive Buffer Register	0024_H	Page 24-39
RXFCON	Receive FIFO Control Register	0030_H	Page 24-40
TXFCON	Transmit FIFO Control Register	0034_H	Page 24-42
FSTAT	FIFO Status Register	0038_H	Page 24-44

The PISEL register controls the input signal selection of the SSC module. Each input of the module kernel receive, transmit and clock signals has associated two input lines (port A and port B).

PISEL
Port Input Select Register **Reset Value: 0000 0000$_H$**

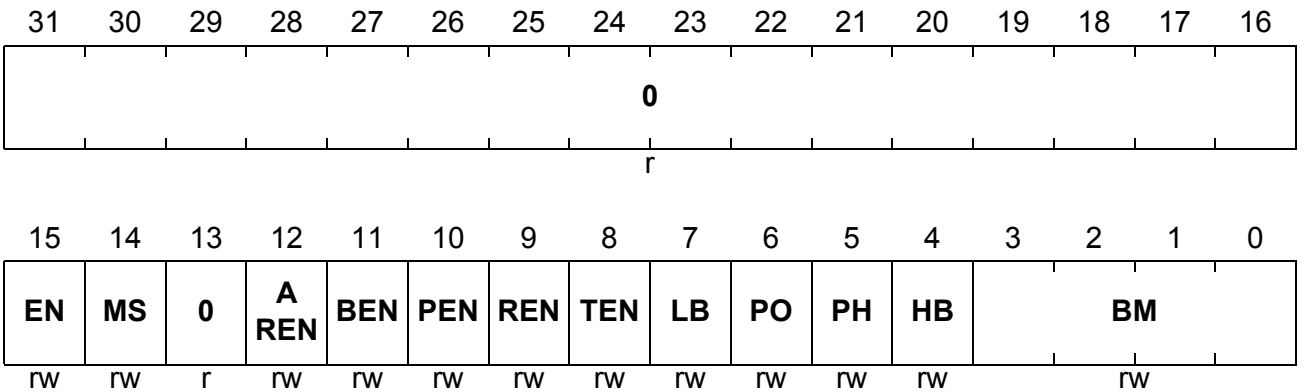


Field	Bits	Type	Description
MRIS	0	rw	**Master Mode Receive Input Select** MRIS selects the receive input line in master mode. 0 Receive input line MRSTA is selected 1 Receive input line MRSTB is selected
SRIS	1	rw	**Slave Mode Receive Input Select** SRIS selects receive input line that in slave mode. 0 Receive input line MTSRA is selected 1 Receive input line MTSRB is selected
SCIS	2	rw	**Slave Mode Clock Input Select** SCIS selects the module kernel SCLK input line that is used as clock input line in slave mode. 0 Slave mode clock input line SCLKA is selected 1 Slave mode clock input line SCLKB is selected
SLSIS	[5:3]	rw	**Slave Mode Slave Select Input Selection** 000_B Slave select input lines are deselected; SSC is operating without slave select input functionality. 001_B $\overline{SLSI1}$ input line is selected for operation 010_B $\overline{SLSI2}$ input line is selected for operation 011_B $\overline{SLSI3}$ input line is selected for operation 100_B $\overline{SLSI4}$ input line is selected for operation 101_B $\overline{SLSI5}$ input line is selected for operation 110_B $\overline{SLSI6}$ input line is selected for operation 111_B $\overline{SLSI7}$ input line is selected for operation

Field	Bits	Type	Description
STIP	8	rw	**Slave Transmit Idle State Polarity** This bit defines the logic level of the slave mode transmit signal MRST when the SSC is deselected (PISEL.SLSIS = 0). 0 MRST = 0 when SSC is deselected in slave mode 1 MRST = 1 when SSC is deselected in slave mode
0	[7:6], [31:9]	r	**Reserved;** read as 0; should be written with 0.

The operating modes of the SSC are controlled by the control register CON. This register contains control bits for mode and error check selection.

CON
Control Register **Reset Value: 0000 0000$_H$**

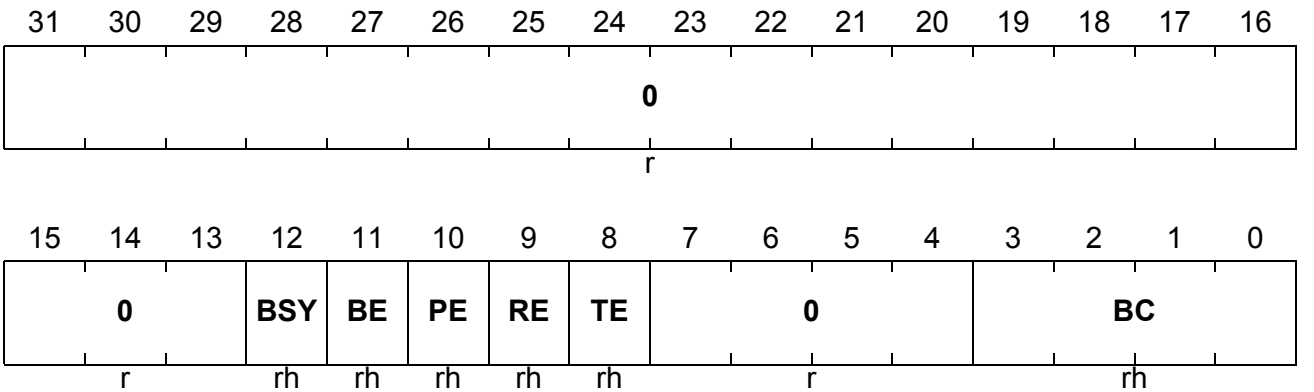


Field	Bits	Type	Description
BM	[3:0]	rw	**Data Width Selection** BM defines the number of data bits of the serial frame. 0000 Reserved; do not use this combination. 0001 to 1111 Transfer Data Width is 2 … 16 bit (<BM> + 1)
HB	4	rw	**Heading Control** 0 Transmit/Receive LSB First 1 Transmit/Receive MSB First
PH	5	rw	**Clock Phase Control** 0 Shift transmit data on the leading clock edge, latch on trailing edge 1 Latch receive data on leading clock edge, shift on trailing edge
PO	6	rw	**Clock Polarity Control** 0 Idle clock line is low, the leading clock edge is low-to-high transition 1 Idle clock line is high, the leading clock edge is high-to-low transition
LB	7	rw	**Loop Back Control** 0 Normal output 1 Receive input is connected with transmit output (Half-duplex Mode)

Field	Bits	Type	Description
TEN	8	rw	**Transmit Error Enable** 0 Ignore transmit errors 1 Check transmit errors
REN	9	rw	**Receive Error Enable** 0 Ignore receive errors 1 Check receive errors
PEN	10	rw	**Phase Error Enable** 0 Ignore phase errors 1 Check phase errors
BEN	11	rw	**Baud Rate Error Enable** 0 Ignore baud rate errors 1 Check baud rate errors
AREN	12	rw	**Automatic Reset Enable** 0 No additional action upon a baud rate error 1 SSC is automatically reset on a baud rate error
MS	14	rw	**Master Select** 0 Slave Mode. Operate on shift clock received via SCLK 1 Master Mode. Generate shift clock and output it via SCLK The inverted state of this bit is available at the M/S select output line.
EN	15	rw	**Enable Bit** 0 Transmission and reception is disabled 1 Transmission and reception is enabled This bit is available at the enable output line.
0	13, [31:16]	r	**Reserved;** read as 0; should be written with 0.

The status register STAT contains status flags for error identification, the busy flag, and a bit field that indicates the current shift counter status.

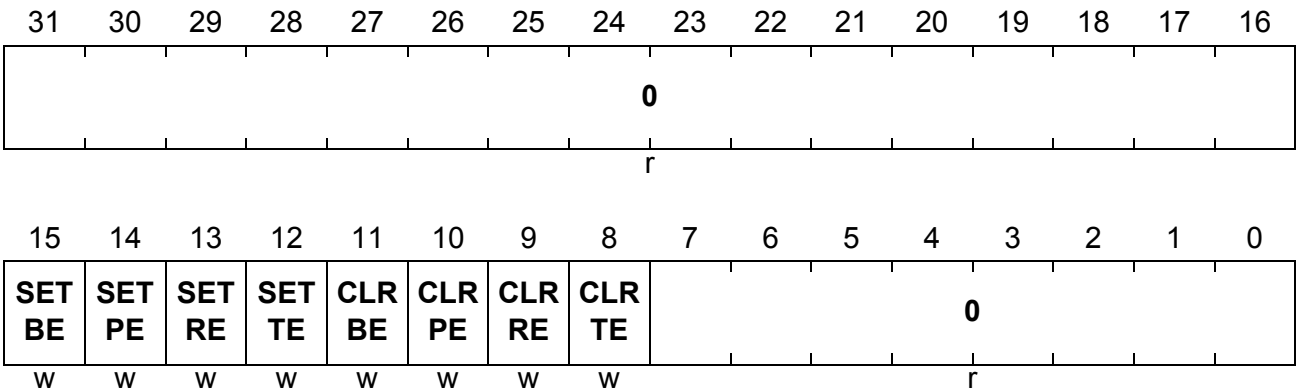
STAT
Status Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
BC	[3:0]	rh	**Bit Count Status** BC indicates the current status of the shift counter. The shift counter is updated with every shifted bit.
TE	8	rh	**Transmit Error Flag** 0 No error 1 Transfer starts with the slave's transmit buffer not being updated
RE	9	rh	**Receive Error Flag** 0 No error 1 Reception completed before the receive buffer was read
PE	10	rh	**Phase Error Flag** 0 No error 1 Received data changes around the sampling clock edge
BE	11	rh	**Baud Rate Error Flag** 0 No error 1 More than factor 2 or 0.5 between slave's actual and expected baud rate
BSY	12	rh	**Busy Flag** BSY is set while a transfer is in progress.
0	[7:4], [31:13]	r	**Reserved;** read as 0; should be written with 0.

The error flag modification register EFM is required for resetting or setting the four error flags, which are located in register CON.

EFM
Error Flag Modification Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CLRTE	8	w	**Clear Transmit Error Flag Bit** 0 No effect 1 Bit CON.TE is cleared Bit is always read as 0.
CLRRE	9	w	**Clear Receive Error Flag Bit** 0 No effect 1 Bit CON.RE is cleared Bit is always read as 0.
CLRPE	10	w	**Clear Phase Error Flag Bit** 0 No effect 1 Bit CON.PE is cleared Bit is always read as 0.
CLRBE	11	w	**Clear Baud Rate Error Flag Bit** 0 No effect 1 Bit CON.BE is cleared Bit is always read as 0.
SETTE	12	w	**Set Transmit Error Flag Bit** 0 No effect 1 Bit CON.TE is set Bit is always read as 0.

Field	Bits	Type	Description
SETRE	13	w	**Set Receive Error Flag Bit** 0 No effect 1 Bit CON.RE is set Bit is always read as 0.
SETPE	14	w	**Set Phase Error Flag Bit** 0 No effect 1 Bit CON.PE is set Bit is always read as 0.
SETBE	15	w	**Set Baud Rate Error Flag Bit** 0 No effect 1 Bit CON.BE is set Bit is always read as 0.
0	[7:0], [31:16]	r	**Reserved;** read as 0; should be written with 0.

Note: When the set and clear bit for an error flag is set at the same time during an EFM write operation (e.g. SETPE = CLRPE = 1), the error flag in STAT is not affected. The chip select control register controls the operation of the chip select generation unit.

The chip select control register controls the operation of the chip select generation unit.

SSOC
Slave Select Output Control Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

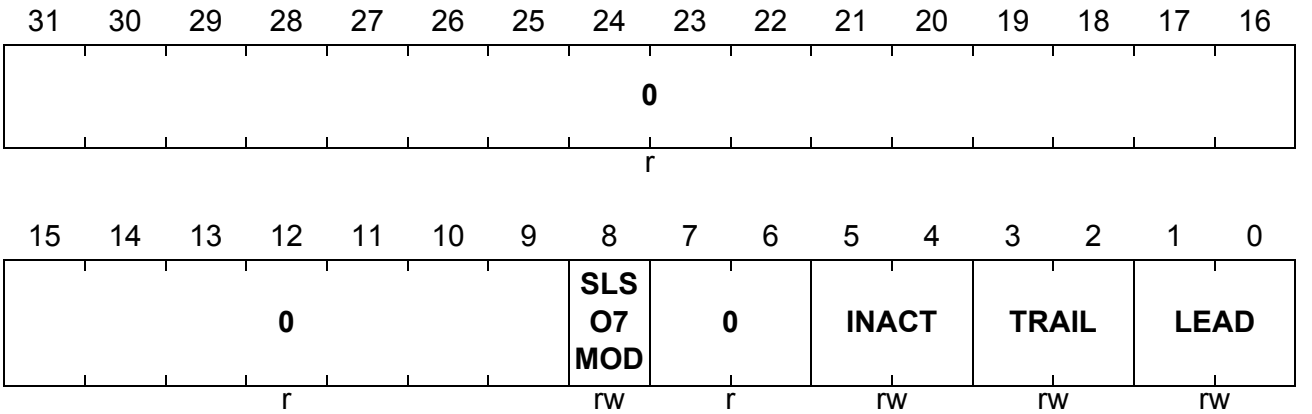
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEN 7	OEN 6	OEN 5	OEN 4	OEN 3	OEN 2	OEN 1	OEN 0	AOL 7	AOL 6	AOL 5	AOL 4	AOL 3	AOL 2	AOL 1	AOL 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
AOLn (n = 0-7)	n	rw	**Active Output Level** 0 SLSOn is at low-level during the chip select active time $t_{SLSOACT}$. The high level is the inactive level of SLSOn. 1 SLSO line n is at high level during the chip select active time $t_{SLSOACT}$. The low-level is the inactive level of SLSOn.
OENn (n = 0-7)	8 + n	rw	**Output n Enable Control** 0 SLSOn output is disabled; SLSOn is always at inactive level as defined by AOLn. 1 SLSOn output is enabled
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

Note: This register is buffered during a transfer.

The chip select control register controls the operation of the chip select generation unit.

SSOTC
Slave Select Output Timing Control Register **Reset Value: 0000 0000$_H$**



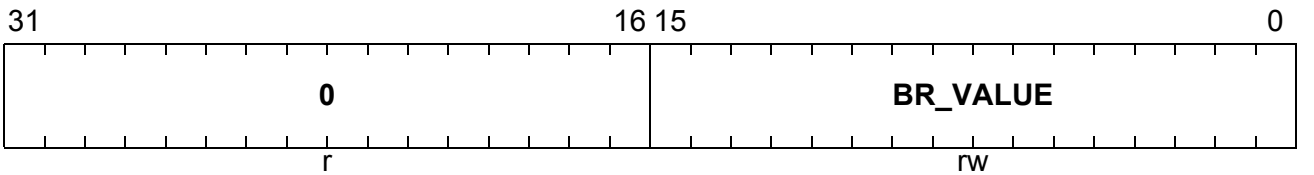
Field	Bits	Type	Description
LEAD	[1:0]	rw	**Slave Output Select Leading Delay** This bit field defines the number of leading. 00 Zero leading delay clock cycles selected 01 One leading delay clock cycle selected 10 Two leading delay clock cycles selected 11 Three leading delay clock cycles selected A leading delay clock cycle is always a multiple of an SCLK shift clock period.
TRAIL	[3:2]	rw	**Slave Output Select Trailing Delay** 00 Zero trailing delay clock cycles selected 01 One trailing delay clock cycle selected 10 Two trailing delay clock cycles selected 11 Three trailing delay clock cycles selected A trailing delay clock cycle is always a multiple of an SCLK shift clock period.
INACT	[5:4]	rw	**Slave Output Select Inactive Delay** 00 Zero inactive delay clock cycles selected 01 One inactive delay clock cycle selected 10 Two inactive delay clock cycles selected 11 Three inactive delay clock cycles selected A inactive delay clock cycle is always a multiple of an SCLK shift clock period.

Field	Bits	Type	Description
SLSO7MOD	8	rw	**SLSO7 Delayed Mode Selection** This bit selects the delayed mode for the SLSO7 slave select output. 0 Normal mode selected for SLSO7 1 Delayed mode selected for SLSO7
0	[7:6], [31:9]	r	**Reserved;** read as 0; should be written with 0.

Note: This register is buffered during a transfer.

The SSC baud rate timer reload register BR contains the 16-bit reload value for the baud rate timer.

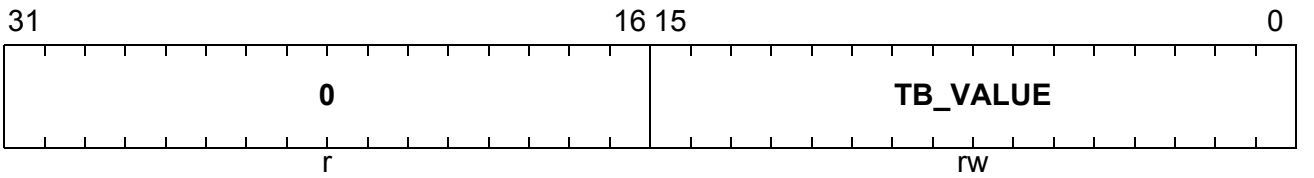
BR
Baud Rate Timer Reload Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
BR_VALUE	[15:0]	rw	**Baud Rate Timer/Reload Register Value** Reading BR returns the 16-bit content of the baud rate timer. Writing BR loads the baud rate timer reload register with BR_VALUE.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

The SSC transmit buffer register TB contains the transmit data value.

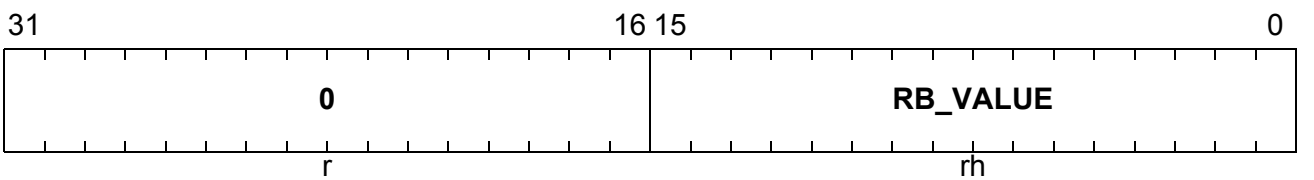
TB
Transmit Buffer Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
TB_VALUE	[15:0]	rw	**Transmit Data Register Value** TB_VALUE is the data value to be transmitted. Unselected bits of TB are ignored during transmission.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

The SSC receive buffer register RB contains the receive data value.

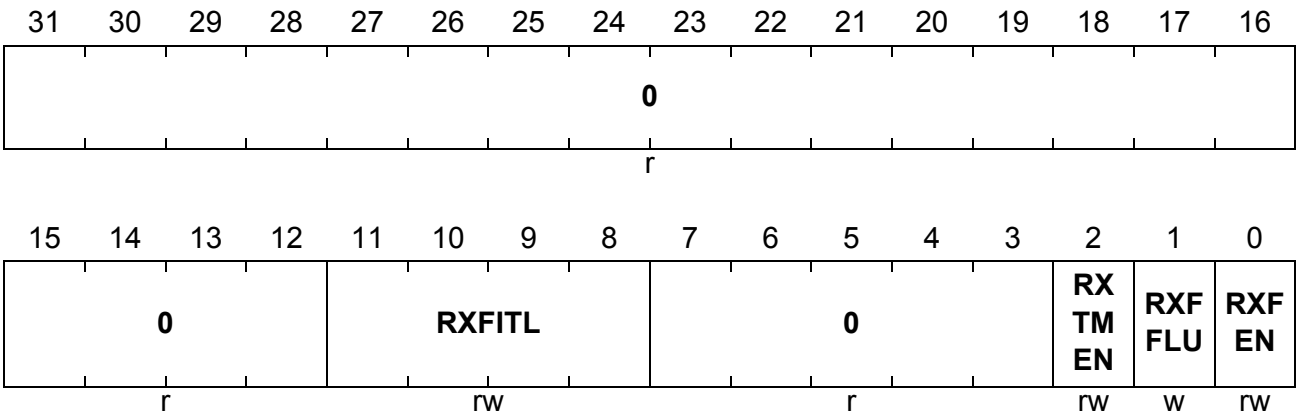
RB
Receive Buffer Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RB_VALUE	[15:0]	rh	**Receive Data Register Value** RB contains the received data value RB_VALUE. Unselected bits of RB will be not valid and should be ignored.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

The receive FIFO control register RXFIFO contains control bits and bit fields that define the operating mode of the receive FIFO.

RXFCON
Receive FIFO Control Register **Reset Value: 0000 0100$_H$**

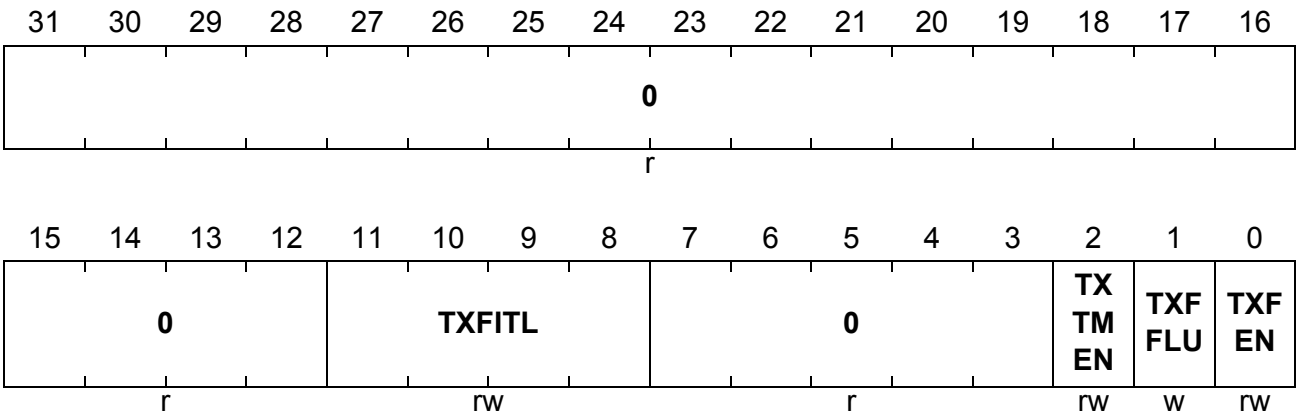


Field	Bits	Type	Description
RXFEN	0	rw	**Receive FIFO Enable** 0 Receive FIFO is disabled 1 Receive FIFO is enabled *Note: Resetting RXFEN automatically flushes the receive FIFO.*
RXFFLU	1	w	**Receive FIFO Flush** 0 No operation 1 Receive FIFO is flushed *Note: Setting RXFFLU clears bit field FSTAT.RXFFL. Bit RXFFLU is always read as 0.*
RXTMEN	2	rw	**Receive FIFO Transparent Mode Enable** 0 Receive FIFO Transparent Mode is disabled 1 Receive FIFO Transparent Mode is enabled *Note: This bit is not applicable if the receive FIFO is disabled (RXFEN = 0).*

Field	Bits	Type	Description
RXFITL	[11:8]	rw	**Receive FIFO Interrupt Trigger Level** Defines the receive FIFO interrupt trigger level. A receive interrupt request (RIR) is always generated after the reception of a byte when the filling level of the receive FIFO is equal to or greater than RXFITL. 0000_B Reserved 0001_B Interrupt trigger level is set to one 0010_B Interrupt trigger level is set to two 0011_B Interrupt trigger level is set to three 0100_B Interrupt trigger level is set to four Others Reserved *Note: In Transparent Mode, this bit field is not applicable.*
0	[7:3], [31:12]	r	**Reserved;** read as 0; should be written with 0.

The transmit FIFO control register TXFIFO contains control bits and bit fields that define the operating mode of the transmit FIFO.

TXFCON
Transmit FIFO Control Register **Reset Value: 0000 0100$_H$**

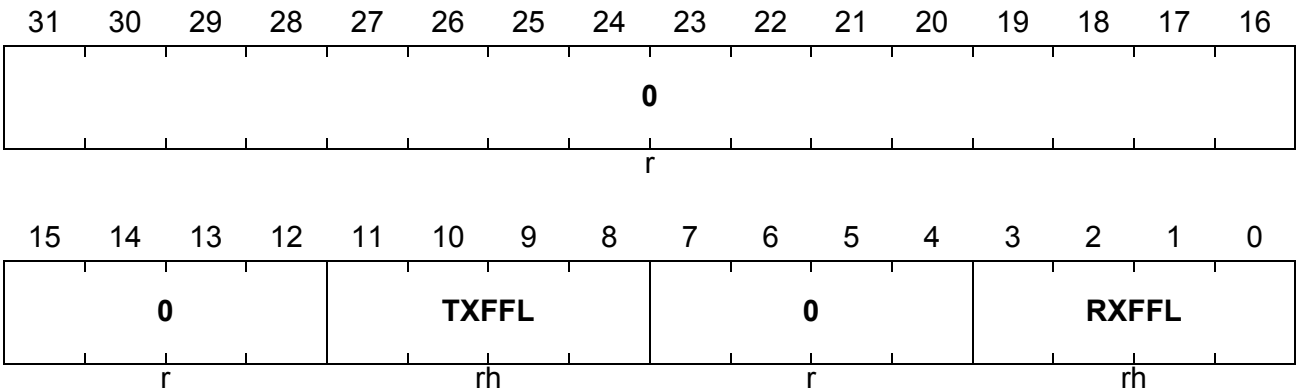


Field	Bits	Type	Description
TXFEN	0	rw	**Transmit FIFO Enable** 0 Transmit FIFO is disabled 1 Transmit FIFO is enabled *Note: Resetting TXFEN automatically flushes the transmit FIFO.*
TXFFLU	1	w	**Transmit FIFO Flush** 0 No operation 1 Transmit FIFO is flushed *Note: Setting TXFFLU clears bit field FSTAT.TXFFL. Bit TXFFLU is always read as 0.*
TXTMEN	2	rw	**Transmit FIFO Transparent Mode Enable** 0 Transmit FIFO Transparent Mode is disabled 1 Transmit FIFO Transparent Mode is enabled *Note: This bit is not applicable if the transmit FIFO is disabled (TXFEN = 0).*

Field	Bits	Type	Description
TXFITL	[11:8]	rw	**Transmit FIFO Interrupt Trigger Level** Defines a transmit FIFO interrupt trigger level. A transmit interrupt request (TIR) is always generated after the transfer of a byte when the filling level of the transmit FIFO is equal to or lower than TXFITL. 0000_B Reserved 0001_B Interrupt trigger level is set to one 0010_B Interrupt trigger level is set to two 0011_B Interrupt trigger level is set to three 0100_B Interrupt trigger level is set to four Others Reserved *Note: In Transparent Mode, this bit field is not applicable.*
0	[7:3], [31:12]	r	**Reserved;** read as 0; should be written with 0.

The FIFO status register FSTAT indicates the filling levels of the receive and transmit FIFOs.

FSTAT
FIFO Status Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RXFFL	[3:0]	rh	**Receive FIFO Filling Level[1)]** This bit field indicates the filling level of the RXFIFO. 0000_B Receive FIFO is filled with zero byte 0001_B Receive FIFO is filled with one byte 0010_B Receive FIFO is filled with two bytes 0011_B Receive FIFO is filled with three bytes 0100_B Receive FIFO is filled with four bytes Others Reserved *Note: RXFFL is cleared after a receive FIFO flush operation.*
TXFFL	[11:8]	rh	**Transmit FIFO Filling Level[1)]** This bit field indicates the filling level of the TXFIFO. 0000_B Transmit FIFO is filled with zero byte 0001_B Transmit FIFO is filled with one byte 0010_B Transmit FIFO is filled with two bytes 0011_B Transmit FIFO is filled with three bytes 0100_B Transmit FIFO is filled with four bytes Others Reserved *Note: TXFFL is cleared after a transmit FIFO flush operation.*
0	[7:4], [31:12]	r	**Reserved;** read as 0; should be written with 0.

1) The data width of a RXFIFO and TXFIFO stage can be programmed from 2 to 15 bits. The data width “byte” mentioned in this description represents a data width of 8 bits.

24.3 SSC0/SSC1 Module Implementation

This section describes SSC0/SSC1 module interfaces with the clock control, port connections, interrupt control, and address decoding.

24.3.1 Interfaces of the SSC Modules

Figure 24-16 shows the TC1130 specific implementation details and interconnections of the SSC0/SSC1 modules. Each of the SSC modules is further supplied by clock control, interrupt control, address decoding, and port control logic. Two DMA requests can be generated by each SSC module.

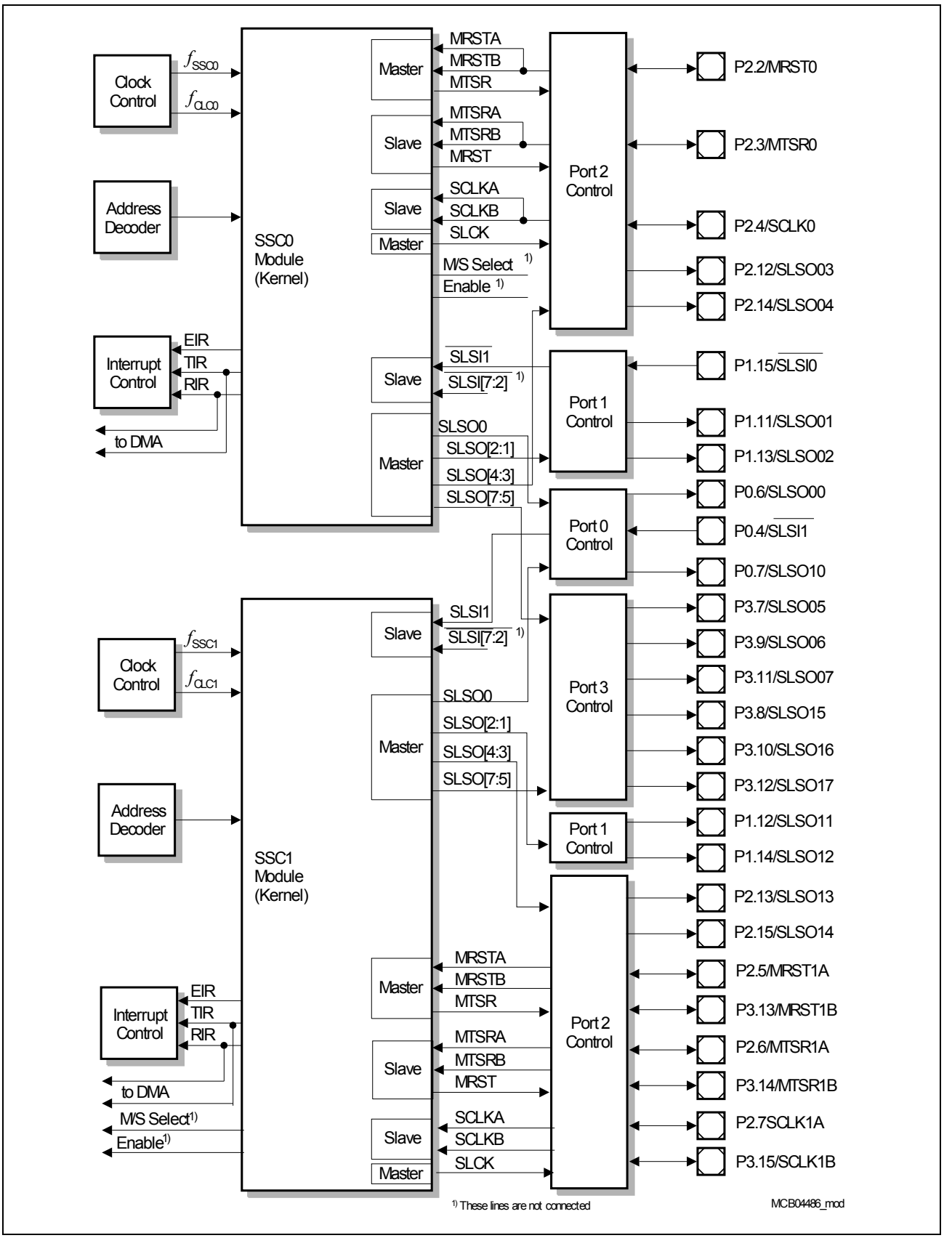


Figure 24-16 SSC0/SSC1 Module Implementation and Interconnections

24.3.2 SSC0/SSC1 Module Related External Registers

Figure 24-17 summarizes the module related external registers, which are required for SSC0/SSC1 programming (see also **Figure 24-15** for the module kernel specific registers).

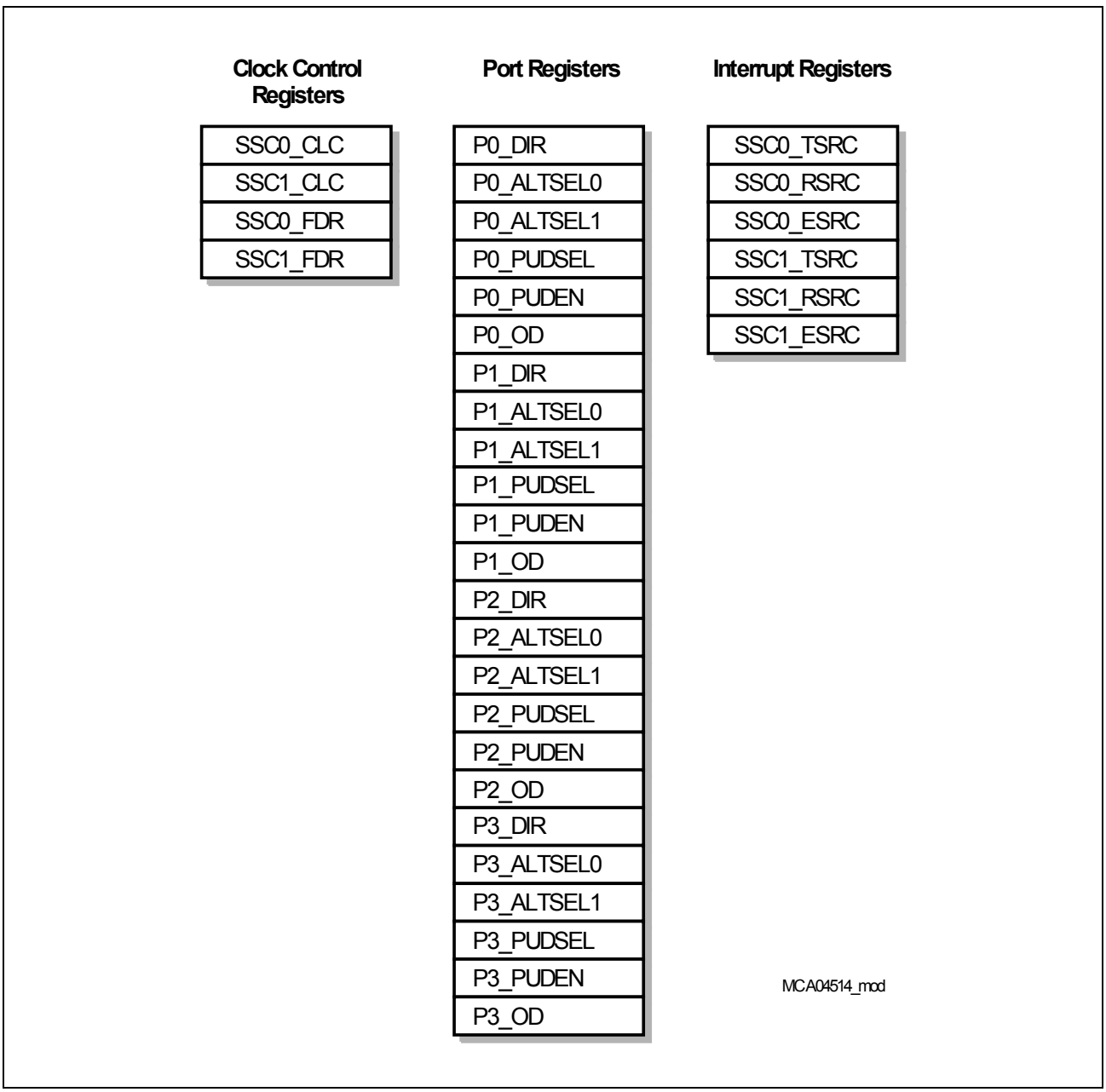


Figure 24-17 SSC0/SSC1 Implementation Specific Special Function Registers

24.3.3 Clock Control

The SSC modules are provided each with two clock signals:

- f_{CLC0} and f_{CLC1}
  This is the module clock that is used inside the SSC kernel for control purposes such as e.g. for clocking of control logic and register operations. The frequency of f_{CLC0} and f_{CLC1} is always identical to the system clock frequency f_{SYS}. The clock control registers SSC0_CLC and SSC1_CLC allow the enabling/disabling of f_{CLC0} and f_{CLC1} under certain conditions.
- f_{SSC0} and f_{SSC1}
  This clock is the module clock that is used in the SSC as input clock of the baud-rate generator which finally defines the baud rate of the serial data. The fractional divider registers SSC0_FDR and SSC1_FDR control the frequency of f_{SSC0} and f_{SSC1} and allow them to be enabled/disabled independently of f_{CLC0} and f_{CLC1}.
  The baud rate timer reload register SSC0_BR and SSC1_BR define serial data baud rate dependent from the frequency of f_{SSC0} and f_{SSC1}.

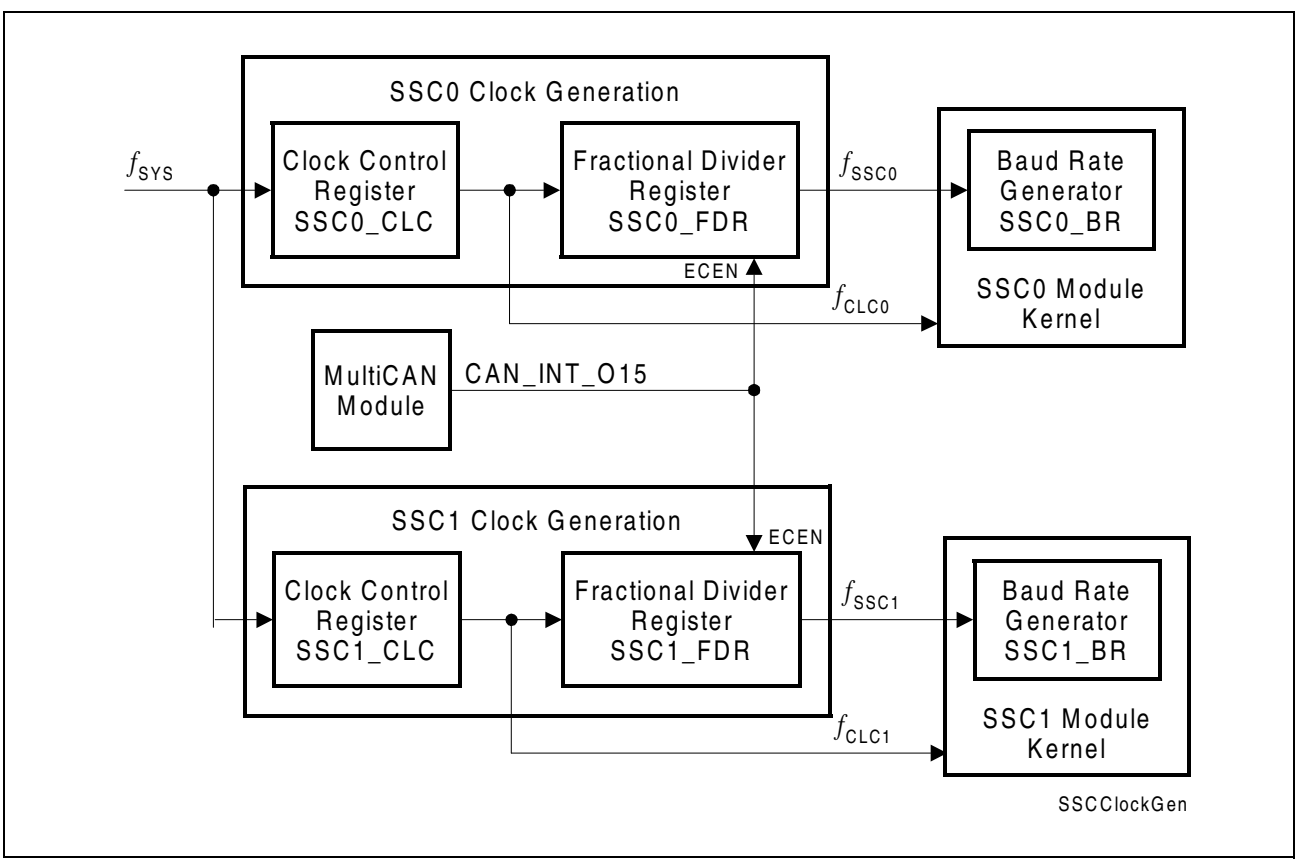


Figure 24-18 SSC Clock Generation

Output signal CAN_INT_O15 of the MultiCAN module can be used for external clock enable control of the fractional divider.

The following formulas define the frequency of f_{SSC0} or f_{SSC1}:

$$f_{SSCx} = f_{SYS} \times \frac{1}{n} \text{ with n = 1024 - FDR.STEP or}$$
$$f_{SSCx} = f_{SYS} \times \frac{n}{1024} \text{ with n = 0 - 1023} \quad (24.2)$$

Note: In SSC master mode, the maximum shift clock frequency is f_{SSCx}/2. In SSC slave mode the maximum shift clock frequency is f_{SSCx}/4.

Combined with the formulas of the baud-rate generator (see **Page 24-19**) and the fractional divider (see chapter “System Control Unit” of the TC1130 System Units User’s Manual), the resulting serial data baud rate is defined by:

$$\text{Baudrate}_{SSC} = \frac{f_{SYS}}{2 \times (\text{BR.BR\_VALUE} + 1) \times (1024 - \text{FDR.STEP})}$$
$$\text{Baudrate}_{SSC} = \frac{f_{SYS} \times \text{FDR.STEP}}{2 \times (\text{BR.BR\_VALUE} + 1) \times 1024} \text{ with FDR.STEP = 0 - 1023} \quad (24.3)$$

Note: The upper formula applies to normal divider mode of the fractional divider (FDR.DM = 01_B). The lower formula applies to fractional divider mode (FDR.DM = 10_B).

The clock control registers allow the control (enable/disable) of the clock signals f_{CLC0} and f_{CLC1} under certain conditions. Each SSC has its own clock control register.

SSC0_CLC
SSC0 Clock Control Register **Reset Value: 0000 0003$_H$**
SSC1_CLC
SSC1 Clock Control Register **Reset Value: 0000 0003$_H$**

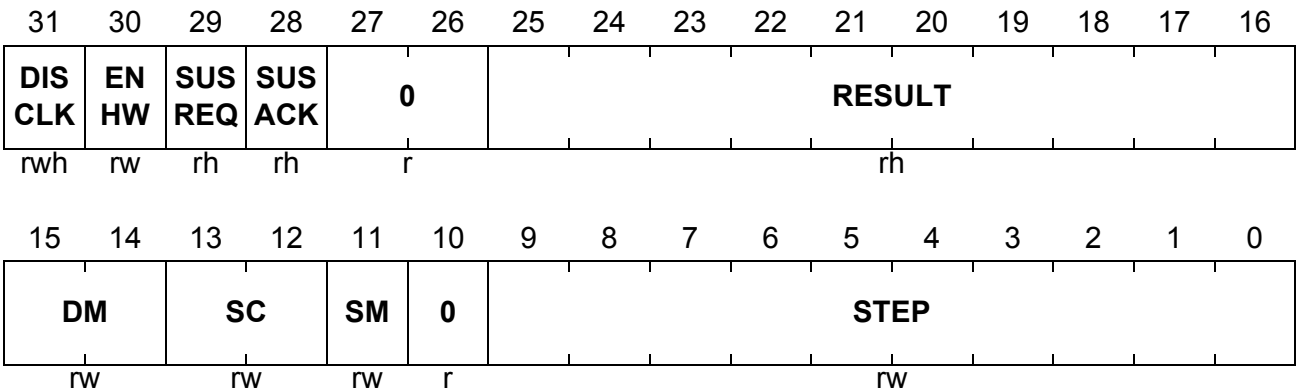
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										FSOE	SBWE	EDIS	SPEN	DISS	DISR
r										rw	w	rw	rw	r	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the module.
SPEN	2	rw	**Module Suspend Enable for OCDS** Used for enabling the suspend mode.
EDIS	3	rw	**External Request Disable** Used for controlling the external clock disable request.
SBWE	4	w	**Module Suspend Bit Write Enable for OCDS** Defines whether SPEN and FSOE are write protected.
FSOE	5	rw	**Fast Switch Off Enable** Used for fast clock switch off in OCDS suspend mode.
0	[31:6]	r	**Reserved;** read as 0; should be written with 0.

The fractional divider register controls the clock rate of the shift clock f_{SSC0} and f_{SSC1}. Each SSC has its own fractional divider register.

SSC0_FDR
SSC0 Fractional Divider Register **Reset Value: 0000 0000$_H$**
SSC1_FDR
SSC1 Fractional Divider Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
STEP	[9:0]	rw	**Step Value** Reload or addition value for RESULT.
SM	11	rw	**Suspend Mode** 0 Granted suspend mode 1 Immediate suspend mode
SC	[13:12]	rw	**Suspend Control** This bit field defines the behavior of the fractional divider in suspend mode.
DM	[15:14]	rw	**Divider Mode** This bit field selects normal divider mode, fractional divider mode, and off-state.
RESULT	[25:16]	rh	**Result Value** Bit field for the addition result.
SUSACK	28	rh	**Suspend Mode Acknowledge** Indicates state of SPNDACK signal.
SUSREQ	29	rh	**Suspend Mode Request** Indicates state of SPND signal.
ENHW	30	rw	**Enable Hardware Clock Control** Controls operation of ECEN input and DISCLK bit.

Field	Bits	Type	Description
DISCLK	31	rwh	**Disable Clock** Hardware controlled disable for f_{OUT} signal.
0	10, [27:26]	rw	**Reserved;** read as 0; should be written with 0.

24.3.3.1 Port Input Select Register

The SSC1 module provides a Peripheral Input Select Register that is used to switch the MRST, MTSR, SCLK input lines of the SSC1 module kernel to either Port 2 or Port 3 as shown in **Figure 24-19**.

*Note: As shown in **Figure 24-19**, the MRST, MTSR and SCLK lines of the SSC1 module can also be output lines. Port line input/output switching is controlled by the input/output control registers DIR.*

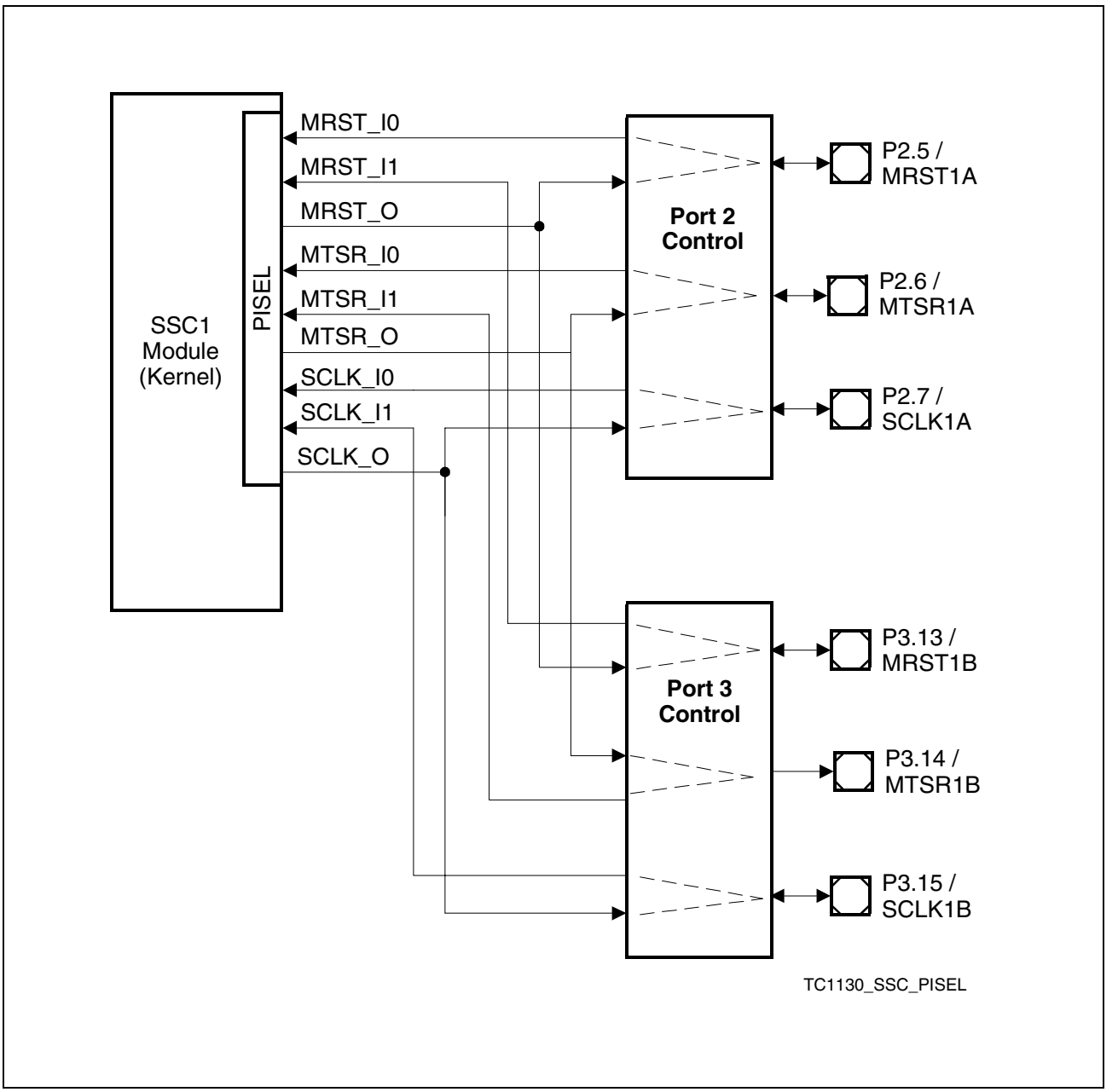


Figure 24-19 Input Line Selection of the SSC1 Module

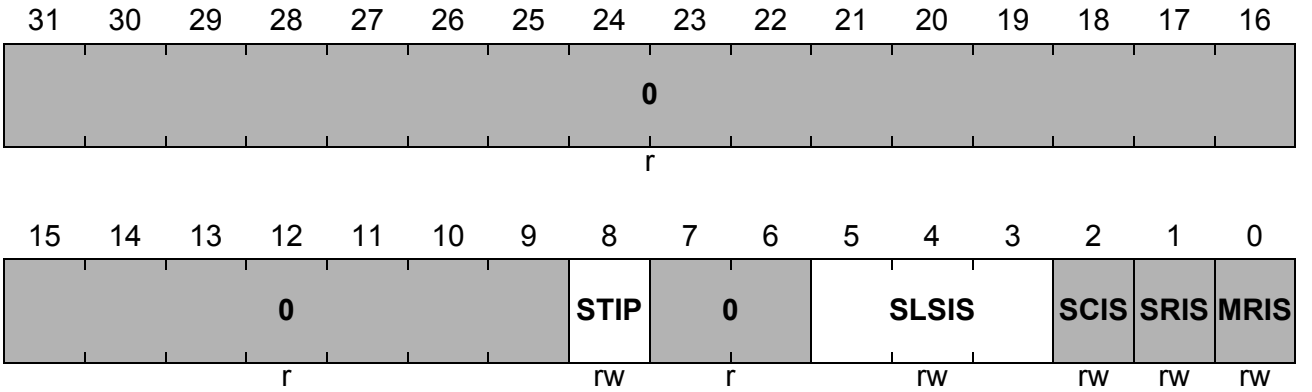
The functionality of the SSC0/SSC1 port input select registers as shown on **Page 24-29** is reduced according the diagram described below:

- No alternate input lines for SSC0 available
- Only one slave select input for each SSC0/SSC1 module used

Therefore, in the port input select registers the corresponding bit fields must be set as required in the application.

SSC0_PISEL
Port Input Select Register

Reset Value: 0000 0000$_H$

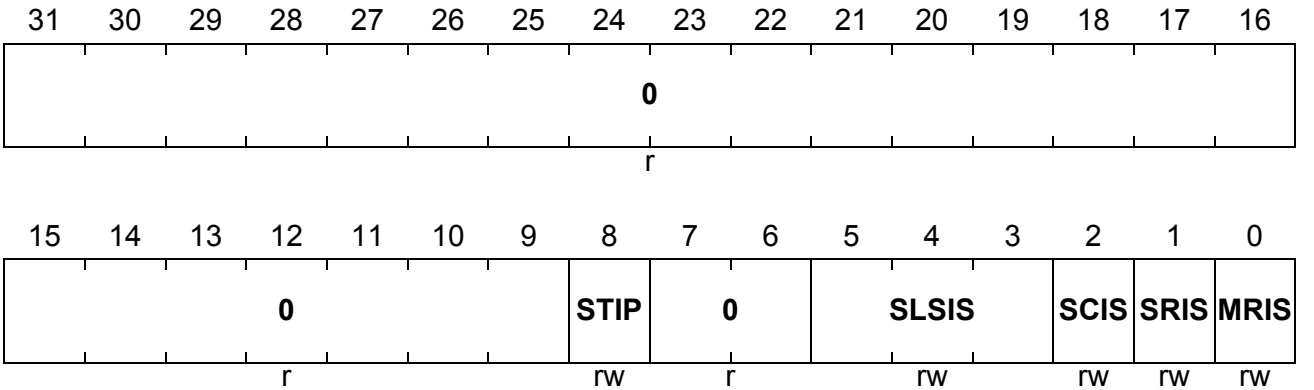


Field	Bits	Type	Description
SLSIS	[5:3]	rw	**Slave Mode Slave Select Input Selection** 000_B Slave select input lines are deselected; SSC is operating without slave select input functionality 001_B SLSI1 input line is selected for operation others: Reserved; do not use these combinations
STIP	8	rw	**Slave Transmit Idle State Polarity** This bit defines the logic level of the slave mode transmit signal MRST when the SSC0 is deselected (PISEL.SLSIS = 0). 0 MRST = 0 when SSC0 is deselected in slave mode 1 MRST = 1 when SSC0 is deselected in slave mode
0	[7:6], [31:9]	r	**Reserved;** read as 0; should be written with 0.

Note: Shaded bits and bit fields are don't care for SSC0 input control.

SSC1_PISEL
Port Input Select Register

Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
MRIS	0	rw	**Master Mode Receive Input Select** MRIS selects the receive input line in master mode. 0 Receive input line MRSTA is selected 1 Receive input line MRSTB is selected
SRIS	1	rw	**Slave Mode Receive Input Select** SRIS selects receive input line that in slave mode. 0 Receive input line MTSRA is selected 1 Receive input line MTSRB is selected
SCIS	2	rw	**Slave Mode Clock Input Select** SCIS selects the module kernel SCLK input line that is used as clock input line in slave mode. 0 Slave mode clock input line SCLKA is selected 1 Slave mode clock input line SCLKB is selected
SLSIS	[5:3]	rw	**Slave Mode Slave Select Input Selection** 000_B Slave select input lines are deselected; SSC is operating without slave select input functionality 001_B SLSI1 input line is selected for operation others: Reserved; do not use these combinations

Field	Bits	Type	Description
STIP	8	rw	**Slave Transmit Idle State Polarity** This bit defines the logic level of the slave mode transmit signal MRST when the SSC1 is deselected (PISEL.SLSIS = 0). 0 MRST = 0 when SSC1 is deselected in slave mode 1 MRST = 1 when SSC1 is deselected in slave mode
0	[7:6], [31:9]	r	**Reserved;** read as 0; should be written with 0.

24.3.3.2 Port Control

The interconnections between the SSC modules and the port I/O lines are controlled in the port logics. The following port control operation selections must be executed (additionally to the PISEL programming):

- Input/output direction selection (DIR registers)
- Alternate function selection (ALTSEL0 and ALTSEL1 registers)
- Input/Output driver characteristic control (PUDSEL, PUDEN and OD registers)

The port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as pull-up/down devices, port direction (input/output), open-drain, and alternate output selections. The I/O lines for the SSC modules are controlled by the port input/output control registers of Port 0, Port 1, Port 2 and Port 3.

Table 24-3 shows how bits and bit fields must be programmed for the required I/O functionality of the SSC I/O lines.

Table 24-3 SSC0 and SSC1 I/O Line Selection and Setup

Module	Port Lines	Input/Output Control Register Bits	I/O
SSC0	P2.2/MRST0	P2_DIR.P2 = 0_B	Input
		P2_DIR.P2 = 1_B	Output
		P2_ALTSEL0.P2 = 1_B	
		P2_ALTSEL1.P2 = 0_B	
	P2.3/MTSR0	P2_DIR.P3 = 0_B	Input
		P2_DIR.P3 = 1_B	Output
		P2_ALTSEL0.P3 = 1_B	
		P2_ALTSEL1.P3 = 0_B	
	P2.4/SCLK0	P2_DIR.P4 = 0_B	Input
		P2_DIR.P4 = 1_B	Output
		P2_ALTSEL0.P4 = 1_B	
		P2_ALTSEL1.P4 = 0_B	
	P1.15/$\overline{\text{SLSI0}}$	P1_DIR.P15 = 0_B	Input

Table 24-3 SSC0 and SSC1 I/O Line Selection and Setup (cont'd)

Module	Port Lines	Input/Output Control Register Bits	I/O
SSC1	P2.5/MRST1A	P2_DIR.P5 = 0_B	Input
		P2_DIR.P5 = 1_B	Output
		P2_ALTSEL0.P5 = 1_B	
		P2_ALTSEL1.P5 = 0_B	
	P3.13/MRST1B	P3_DIR.P13 = 0_B	Input
		P3_DIR.P13 = 1_B	Output
		P3_ALTSEL0.P13 = 1_B	
		P3_ALTSEL1.P13 = 0_B	
	P2.6/MTSR1A	P2_DIR.P6 = 0_B	Input
		P2_DIR.P6 = 1_B	Output
		P2_ALTSEL0.P6 = 1_B	
		P2_ALTSEL1.P6 = 0_B	
	P3.14/MTSR1B	P3_DIR.P14 = 0_B	Input
		P3_DIR.P14 = 1_B	Output
		P3_ALTSEL0.P14 = 1_B	
		P3_ALTSEL1.P14 = 0_B	
	P2.7/SCLK1A	P2_DIR.P7 = 0_B	Input
		P2_DIR.P7 = 1_B	Output
		P2_ALTSEL0.P7 = 1_B	
		P2_ALTSEL1.P7 = 0_B	
	P3.15/SCLK1B	P3_DIR.P15 = 0_B	Input
		P3_DIR.P15 = 1_B	Output
		P3_ALTSEL0.P15 = 1_B	
		P3_ALTSEL1.P15 = 0_B	
	P0.4/$\overline{\text{SLSI1}}$	P0_DIR.P4 = 0_B	Input

Table 24-3 SSC0 and SSC1 I/O Line Selection and Setup (cont'd)

Module	Port Lines	Input/Output Control Register Bits	I/O
Slave Select Outputs			
SSC0	P0.6/SLSO00	P0_DIR.P6 = 1_B	Output
		P0_ALTSEL0.P6 = 1_B	
		P0_ALTSEL1.P6 = 1_B	
	P1.11/SLSO01	P1_DIR.P11 = 1_B	Output
		P1_ALTSEL0.P11 = 0_B	
		P1_ALTSEL1.P11 = 1_B	
	P1.13/SLSO02	P1_DIR.P13 = 1_B	Output
		P1_ALTSEL0.P13 = 0_B	
		P1_ALTSEL1.P13 = 1_B	
	P2.12/SLSO03	P2_DIR.P12 = 1_B	Output
		P2_ALTSEL0.P12 = 0_B	
		P2_ALTSEL1.P12 = 1_B	
	P2.14/SLSO04	P2_DIR.P14 = 1_B	Output
		P2_ALTSEL0.P14 = 0_B	
		P2_ALTSEL1.P14 = 1_B	
	P3.7/SLSO05	P3_DIR.P7 = 1_B	Output
		P3_ALTSEL0.P7 = 1_B	
		P3_ALTSEL1.P7 = 0_B	
	P3.9/SLSO06	P3_DIR.P9 = 1_B	Output
		P3_ALTSEL0.P9 = 1_B	
		P3_ALTSEL1.P9 = 0_B	
	P3.11/SLSO07	P3_DIR.P11 = 1_B	Output
		P3_ALTSEL0.P11 = 1_B	
		P3_ALTSEL1.P11 = 0_B	

Table 24-3 SSC0 and SSC1 I/O Line Selection and Setup (cont'd)

Module	Port Lines	Input/Output Control Register Bits	I/O
SSC1	P0.7/SLSO10	P0_DIR.P7 = 1_B	Output
		P0_ALTSEL0.P7 = 1_B	
		P0_ALTSEL1.P7 = 1_B	
	P1.12/SLSO11	P1_DIR.P12 = 1_B	Output
		P1_ALTSEL0.P12 = 0_B	
		P1_ALTSEL1.P12 = 1_B	
	P1.14/SLSO12	P1_DIR.P14 = 1_B	Output
		P1_ALTSEL0.P14 = 0_B	
		P1_ALTSEL1.P14 = 1_B	
	P2.13/SLSO13	P2_DIR.P13 = 1_B	Output
		P2_ALTSEL0.P13 = 0_B	
		P2_ALTSEL1.P13 = 1_B	
	P2.15/SLSO14	P2_DIR.P15 = 1_B	Output
		P2_ALTSEL0.P15 = 0_B	
		P2_ALTSEL1.P15 = 1_B	
	P3.8/SLSO15	P3_DIR.P8 = 1_B	Output
		P3_ALTSEL0.P8 = 1_B	
		P3_ALTSEL1.P8 = 0_B	
	P3.10/SLSO16	P3_DIR.P10 = 1_B	Output
		P3_ALTSEL0.P10 = 1_B	
		P3_ALTSEL1.P10 = 0_B	
	P3.12/SLSO17	P3_DIR.P12 = 1_B	Output
		P3_ALTSEL0.P12 = 1_B	
		P3_ALTSEL1.P12 = 0_B	

P0_DIR
Port 0 Direction Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 4, 6, 7)**	n	rw	**Port 0 Pin 4, 6, 7 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are not taken into account for SSC I/O port control.

P1_DIR
Port 1 Direction Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 11-15)**	n	rw	**Port 1 Pin 11-15 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for SSC I/O port control.

P2_DIR
Port 2 Direction Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 2-7, 12-15)**	n	rw	**Port 2 Pin 2-7, 12-15 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for SSC I/O port control.

P3_DIR
Port 3 Direction Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 7-15)**	n	rw	**Port 3 Pin 7-15 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for SSC I/O port control.

P0_ALTSELn (n = 0, 1)
Port 0 Alternate Select Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 24-4 Function of the Bits P0_ALTSEL0.Pn and P0_ALTSEL1.Pn (n = 6, 7)[1)]

P0_ALTSEL0.Pn	P0_ALTSEL1.Pn	Function
1	1	Alternate Select 3

1) Shaded bits and bit field are don't care for SSC I/O port control.

P1_ALTSELn (n = 0, 1)
Port 1 Alternate Select Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 24-5 Function of the Bits P1_ALTSEL0.Pn and P1_ALTSEL1.Pn (n = 11-14)[1)]

P1_ALTSEL0.Pn	P1_ALTSEL1.Pn	Function
0	1	Alternate Select 2

1) Shaded bits and bit field are don't care for SSC I/O port control.

P2_ALTSELn (n = 0, 1)
Port 2 Alternate Select Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 24-6 Function of the Bits P2_ALTSEL0.Pn and P2_ALTSEL1.Pn (n = 2-7, 12-15)[1)]

P2_ALTSEL0.Pn	P2_ALTSEL1.Pn	Function
1	0	Alternate Select 1
0	1	Alternate Select 2

1) Shaded bits and bit field are don't care for SSC I/O port control.

P3_ALTSELn (n = 0, 1)
Port 3 Alternate Select Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 24-7 Function of the Bits P3_ALTSEL0.Pn and P3_ALTSEL1.Pn (n = 7-15)[1)]

P3_ALTSEL0.Pn	P3_ALTSEL1.Pn	Function
1	0	Alternate Select 1

1) Shaded bits and bit field are don't care for SSC I/O port control.

The SSC0/SSC1 ports also offer the possibility to configure the following output characteristics:

- Push/pull (optional pull-up/pull-down)
- Open drain with internal pull-up
- Open drain with external pull-up

P0_PUDSEL
Port 0 Pull-Up/Pull-Down Select Register **Reset Value: 0000 FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 4, 6, 7)**	n	rw	**Pull-Up/Pull-Down Select Port 0 Bit n**[1)] 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for SSC I/O port control.

P1_PUDSEL
Port 1 Pull-Up/Pull-Down Select Register **Reset Value: 0000 FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 11-15)**	n	rw	**Pull-Up/Pull-Down Select Port 1 Bit n**[1)] 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for SSC I/O port control.

P2_PUDSEL
Port 2 Pull-Up/Pull-Down Select Register **Reset Value: 0000 FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
r				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 2-7)**	n	rw	**Pull-Up/Pull-Down Select Port 2 Bit n**[1)] 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:12]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for SSC I/O port control.

P3_PUDSEL
Port 3 Pull-Up/Pull-Down Select Register

Reset Value: 0000 FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 7-15)**	n	rw	**Pull-Up/Pull-Down Select Port 3 Bit n**[1)] 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for SSC I/O port control.

P0_PUDEN
Port 0 Pull-Up/Pull-Down Enable Register

Reset Value: 0000 FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

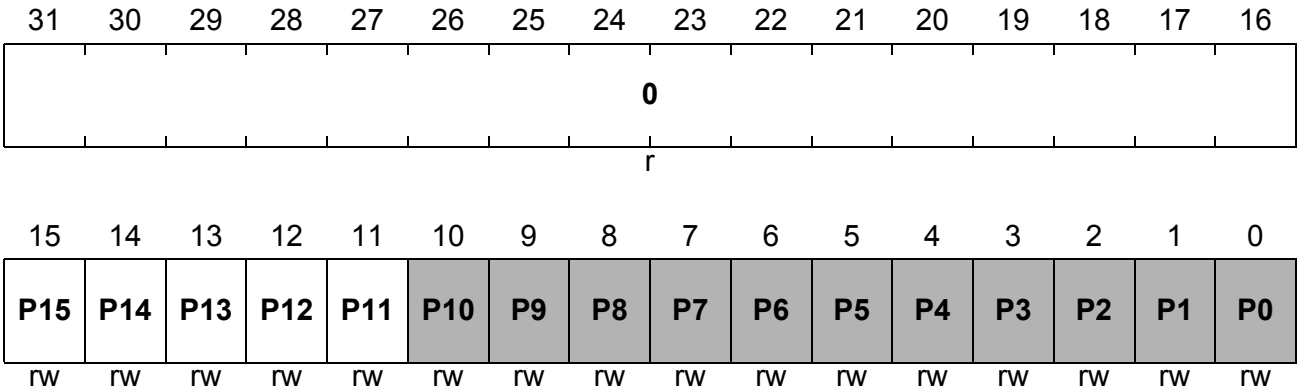
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 4, 6, 7)**	n	rw	**Pull-Up/Pull-Down Enable at Port 0 Bit n**[1)] 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for SSC I/O port control.

P1_PUDEN
Port 1 Pull-Up/Pull-Down Enable Register

Reset Value: 0000 FFFF$_H$

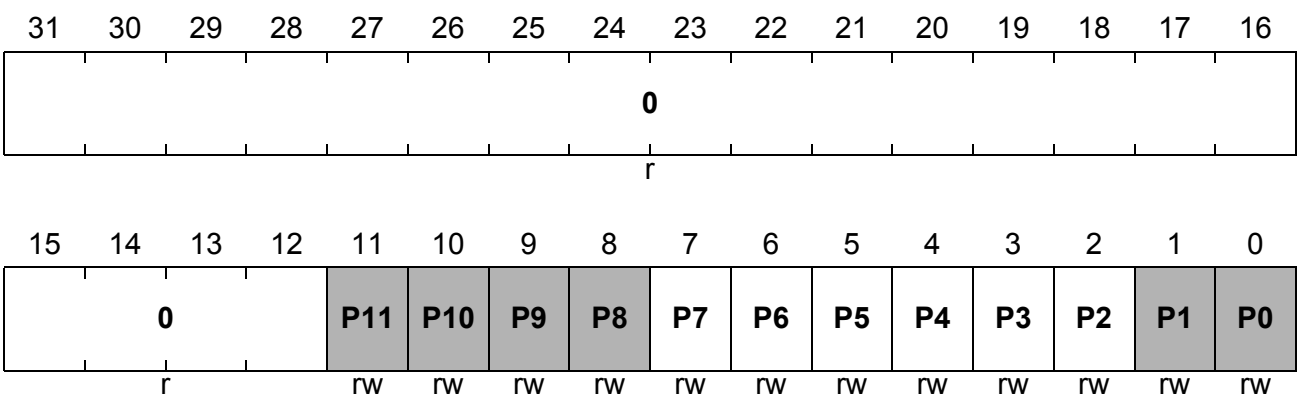


Field	Bits	Type	Description
Pn **(n = 11-15)**	n	rw	**Pull-Up/Pull-Down Enable at Port 1 Bit n**[1)] 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for SSC I/O port control.

P2_PUDEN
Port 2 Pull-Up/Pull-Down Enable Register

Reset Value: 0000 0FFF$_H$



Field	Bits	Type	Description
Pn **(n = 2-7)**	n	rw	**Pull-Up/Pull-Down Enable at Port 2 Bit n**[1)] 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:12]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for SSC I/O port control.

P3_PUDEN
Port 3 Pull-Up/Pull-Down Enable Register

Reset Value: 0000 FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 7-15)**	n	rw	**Pull-Up/Pull-Down Enable at Port 3 Bit n**[1)] 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for SSC I/O port control.

P0_OD
Port 0 Open Drain Control Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 6, 7)**	n	rw	**Port 0 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for SSC I/O port control.

P1_OD
Port 1 Open Drain Control Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 11-14)	n	rw	**Port 1 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for SSC I/O port control.

P2_OD
Port 2 Open Drain Control Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn (n = 2-7, 12-15)	n	rw	**Port 2 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for SSC I/O port control.

P3_OD
Port 3 Open Drain Control Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 7-15)**	n	rw	**Port 3 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for SSC I/O port control.

24.3.3.3 Interrupt Registers

The 2 × 3 interrupts of the SSC0 and SSC1 module are controlled by the following service request control registers:

- SSC0_TSRC, SSC1_TSRC controls the transmit interrupts
- SSC0_RSRC, SSC1_RSRC controls the receive interrupts
- SSC0_ESRC, SSC1_ESRC controls the error interrupts

SSC0_TSRC
SSC0 Transmit Interrupt Service Request Control Register
SSC0_RSRC
SSC0 Receive Interrupt Service Request Control Register
SSC0_ESRC
SSC0 Error Interrupt Service Request Control Register
SSC1_TSRC
SSC1 Transmit Interrupt Service Request Control Register
SSC1_RSRC
SSC1 Receive Interrupt Service Request Control Register
SSC1_ESRC
SSC1 Error Interrupt Service Request Control Register

Reset Values: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETR	CLRR	SRR	SRE	0	TOS	0		SRPN							
w	w	rh	rw	r	rw	r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number**
TOS	10	rw	**Type of Service Control**
SRE	12	rw	**Service Request Enable**
SRR	13	rh	**Service Request Flag**
CLRR	14	w	**Request Clear Bit**
SETR	15	w	**Request Set Bit**

Field	Bits	Type	Description
0	[9:8], 11, [31:16]	r	**Reserved;** read as 0; should be written with 0.

Note: Further details on interrupt handling and processing are described in chapter "Interrupt System" of the TC1130 System Units User's Manual.

24.3.4 DMA Requests

The DMA request lines of the SSC0/SSC1 modules become active whenever its related interrupt line is activated. The DMA request lines are connected to the DMA controller as shown in **Table 24-8**.

Table 24-8 DMA Request Lines of SSC0/SSC1

Module	Related SSC Interrupt	DMA Request Line	Description
SSC0	RIR	SSC0_RDR	SSC0 Receive DMA Request
	TIR	SSC0_TDR	SSC0 Transmit DMA Request
SSC1	RIR	SSC1_RDR	SSC1 Receive DMA Request
	TIR	SSC1_TDR	SSC1 Transmit DMA Request

24.3.5 SSC0/SSC1 Register Address Ranges

In the TC1130, the registers of the two SSC modules are located in the following address ranges:

- SSC0 module: Module Base Address = F010 0100_H
  Module End Address = F010 $01FF_H$
- SSC1 module: Module Base Address = F010 0200_H
  Module End Address = F010 $02FF_H$
- Absolute Register Address = Module Base Address + Offset Address
  (offset addresses see **Table 24-2**)

Note: The complete and detailed address map of the SSC0/SSC1 modules is described in the chapter "Register Overview" of the TC1130 System Units User's Manual.

25 IIC

This chapter describes the IIC Module of the TC1130. It contains the following sections:

- Functional description of the IIC Kernel (see **Section 25.1**)
- Register descriptions of all IIC Kernel specific registers (see **Section 25.2**)
- TC1130 implementation specific details and registers of the IIC (port connections and control, interrupt control, address decoding, clock control, see **Section 25.3**)

*Note: The IIC kernel register names described in **Section 25.2** will be referenced in other parts of the TC1130 User's Manual with the module name prefix "IIC_".*

25.1 IIC Kernel Description

IIC supports a certain protocol to allow devices to communicate directly with each other via two wires. One line is responsible for clock transfer and synchronization (SCL), the other is responsible for the data transfer (SDA).

25.1.1 Introduction

The on-chip IIC Bus module connects the platform buses to other external controllers and/or peripherals via the two-line serial IIC interface. The IIC Bus module provides communication at data rates of up to 400 kbit/s and features 7-bit addressing as well as 10-bit addressing. This module is fully compatible to the IIC bus protocol.

The module can operate in three different modes:

Master Mode, where the IIC controls the bus transactions and provides the clock signal.

Slave Mode, where an external master controls the bus transactions and provides the clock signal.

Multimaster Mode, where several masters can be connected to the bus, i.e. the IIC can be master or slave.

The on-chip IIC bus module allows efficient communication via the common IIC bus. The module unloads the CPU of low level tasks such as:

- (De)Serialization of bus data
- Generation of start and stop conditions
- Monitoring the bus lines in slave mode
- Evaluation of the device address in slave mode
- Bus access arbitration in multimaster mode

Features

- Software compatible to V1.0 of C161RI
- Extended buffer allows up to 4 send/receive data bytes to be stored
- Selectable baud rate generation
- Support of standard 100 kBaud and extended 400 kBaud data rates
- Operation in 7-bit addressing mode or 10-bit addressing mode
- Flexible control via interrupt service routines or by polling
- Dynamic access to up to 2 physical IIC buses

Applications

- EEPROMs
- 7-Segment Displays
- Keyboard Controllers
- On-Screen Display
- Audio Processors

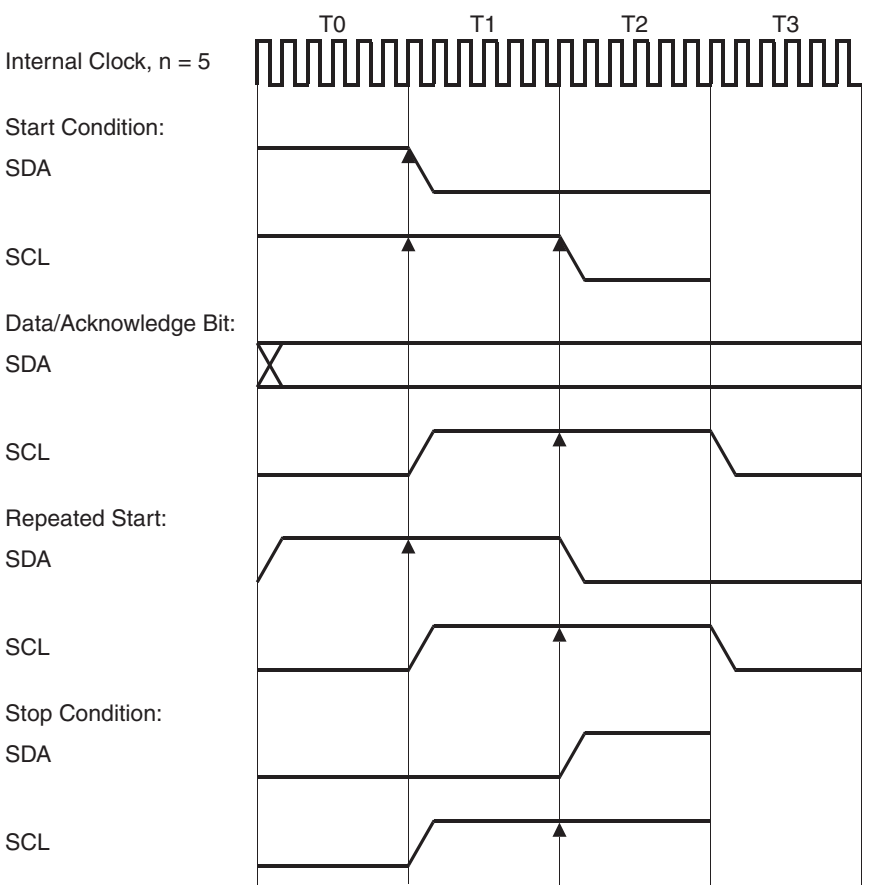
25.1.2 Operational Overview

Data is transferred by the 2-line IIC bus (SDA, SCL) using a protocol that ensures reliable and efficient transfers. This protocol clearly distinguishes regular data transfers from defined control signals which control the data transfers.

The following bus conditions are defined:

- **Bus Idle:** SDA and SCL remain high. The IIC bus is currently not used.
- **Data Valid:** SDA stable during the high phase of SCL. SDA then represents the transferred bit. There is one clock pulse for each transferred bit of data. During data transfers SDA may only change while SCL is low (see below)!
- **Start Transfer:** A falling edge on SDA (↘) while SCL is high indicates a start condition. This start condition initiates a data transfer over the IIC bus.
- **Stop Transfer:** A rising edge on SDA (↗) while SCL is high indicates a stop condition. This stop condition terminates a data transfer. Between a start condition and a stop condition an arbitrary number of bytes may be transferred.

Figure 25-1 below gives examples for these bus conditions.



The high level of the signal is verified. If it is low, Ti is repeated.
T0 ... T3 each Ti has a length of 1 ... 64 internal clocks as defined in ICBD0 and ICBD1.

UED08582_mod

Figure 25-1 Bus Conditions

The Physical IIC Bus Interface

Communication via the IIC Bus uses two bidirectional lines, the serial data line SDA and the serial clock line SCL. These two generic interface lines can each be connected to a number of I/O port lines. These connections can be established and released under software control.

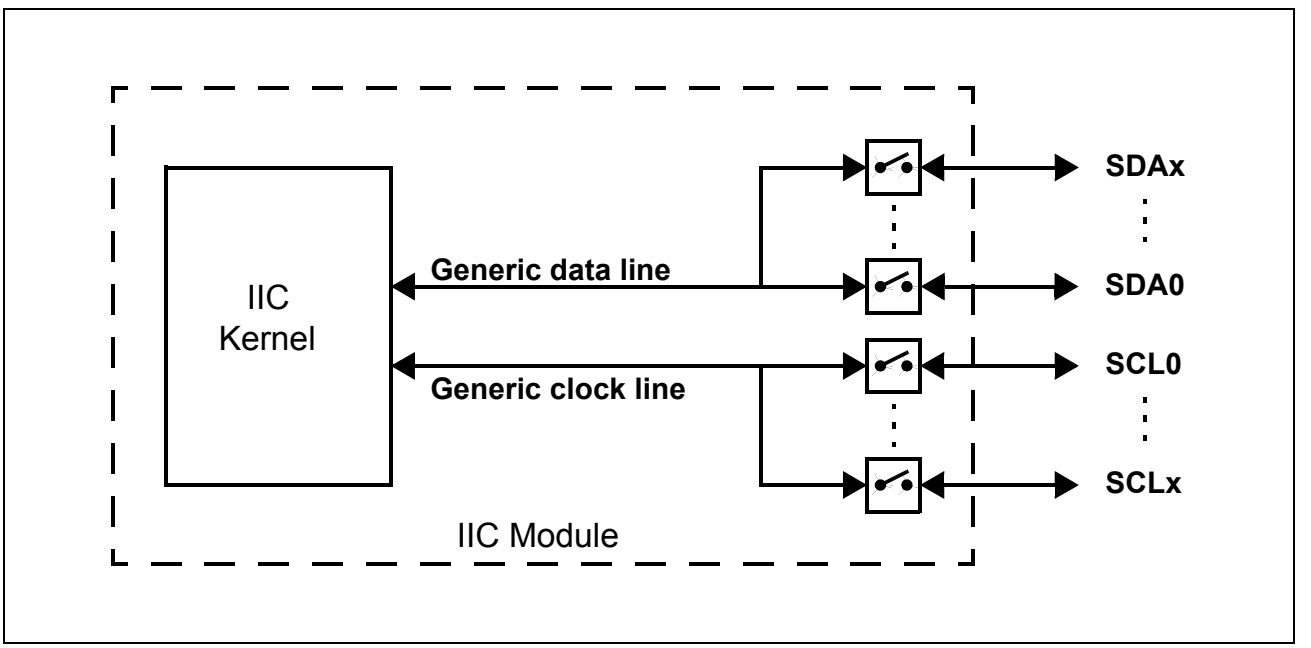


Figure 25-2 IIC Bus Line Connections

This mechanism allows a number of configurations of the physical IIC Bus interface:

Channel switching: The IIC module can be connected to a specific pair of pins (e.g. SDA0 and SCL0) which then forms a separate IIC channel to the external system. The channel can be dynamically switched by connecting the module to another pair of pins (e.g. SDA1 and SCL1). This establishes physically separate interface channels.

Broadcasting: Connecting the module to more than one pair of pins (e.g. SDA0/1 and SCL0/1) allows the transmission of messages over multiple physical channels at the same time. Please note that this configuration is critical when the IIC is a slave. In master mode it cannot be guaranteed that all selected slaves have reached the message.

Register BUSCON selects the bus baud rate as well the activation of SDA and SCL lines. So an external IIC channel can be established (baud rate and physical lines) with one single register access.

Physical channels can be selected, so the IIC module can use electrically separated channels or increase the addressing range by using more data lines.

Note: Baud rate and physical channels should never be changed (via BUSCON) during a transfer.

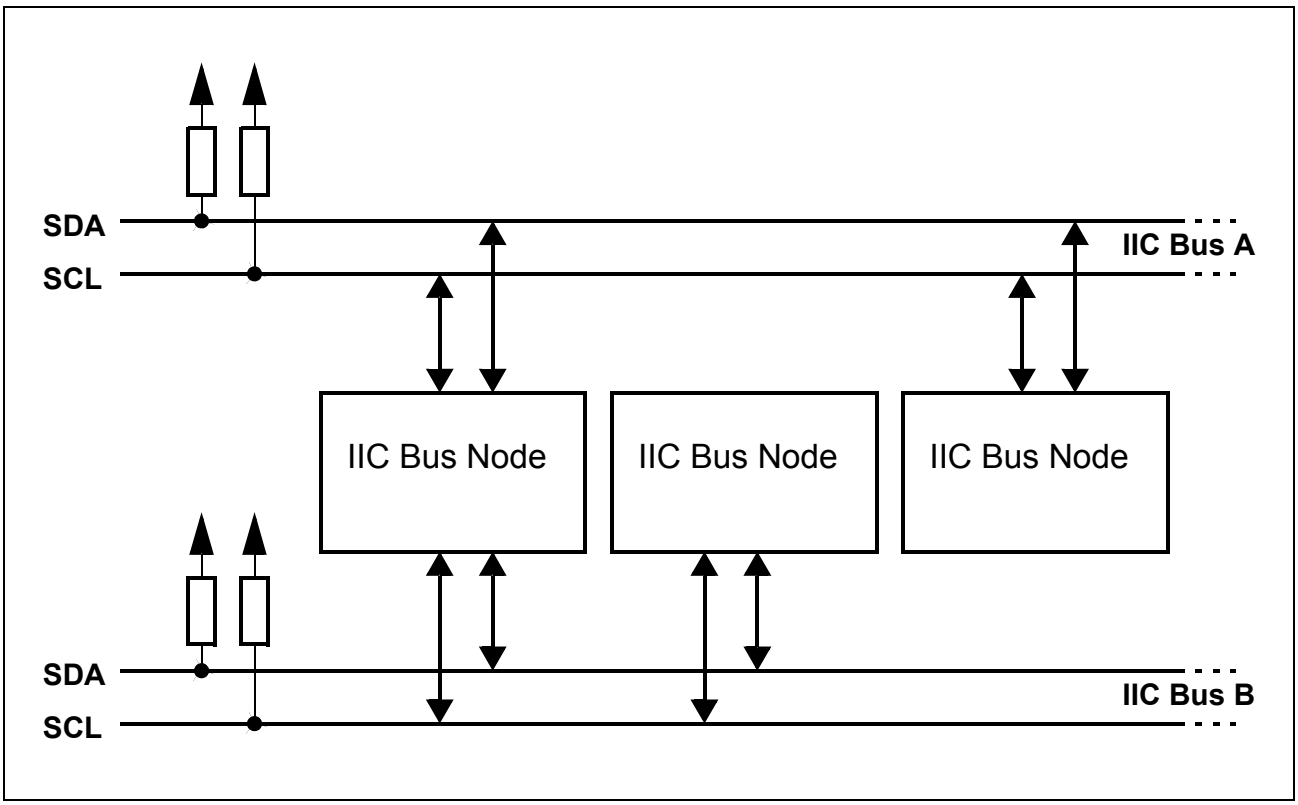


Figure 25-3 Physical Bus Configuration Example

Output Pin Configuration

The pin drivers that are assigned to the IIC channel(s) provide open drain outputs (i.e. no upper transistor). This ensures that the IIC module does not put any load on the IIC bus lines while the IIC is not powered. The IIC bus lines therefore require external pull-up resistors (approx. 10 kΩ for operation at 100 kBaud, 2 kΩ for operation at 400 kBaud).

All pins of the IIC that are to be used for IIC bus communication must be switched to output and their alternate function must be enabled (by setting the respective port output latch to 1), before any communication can be established.

If not driven by the IIC module (i.e. the corresponding enable bit in register BUSCON is 0) they then switch off their drivers (i.e. driving 1 to an open drain output). Due to the external pull-up devices, the respective bus levels will then be 1 which is idle.

The IIC module features digital input filters in order to improve the rejection of noise from the external bus lines.

25.1.3 Functional Overview

25.1.3.1 Operation in Master Mode

If the on-chip IIC module shall control the IIC bus (i.e. be bus master) master mode must be selected via bit field MOD in register SYSCON. The physical channel is configured by a control word written to register BUSCON, defining the active interface pins and the used baud rate. More than one SDA and/or SCL line may be active at a time. The address of the remote slave that is to be accessed is written to RTB0 … 3. The bus is claimed by setting bit BUM in register SYSCON. This generates a start condition on the bus and automatically starts the transmission of the address in RTB0. Bit TRX in register SYSCON defines the transfer direction (TRX = 1, i.e. transmit, for the slave address). A repeated start condition is generated by setting bit RSC in register SYSCON, which automatically starts the transmission of the address previously written to RTB0. This may be used to change the transfer direction. RSC is cleared automatically after the repeated start condition has been generated.

The bus is released by clearing bit BUM in register SYSCON. This generates a stop condition on the bus.

In receive mode, if a data transfer is stopped by setting the STP bit in SYSCON register, no acknowledge is issued when the last byte is shifted in. But if ACKDIS is supposed to disable the acknowledge, the ACKDIS bit must be set before TRX is clear to 0.

25.1.3.2 Operation in Multimaster Mode

If multimaster mode is selected via bit field MOD in register SYSCON the on-chip IIC module can operate concurrently as a bus master or as a slave. The descriptions of these modes apply accordingly.

Multimaster mode implies that several masters are connected to the same bus. As more than one master may try to claim the bus at a given time an arbitration is done on the SDA line. When a master device detects a mismatch between the data bit to be sent and the actual level on the SDA (bus) line it looses the arbitration and automatically switches to slave mode (leaving the other device as the remaining master). This loss of arbitration is indicated by bit AL in register SYSCON, which must be checked by the driver software when operating in multimaster mode. Lost arbitration is also indicated when the software tries to claim the bus (by setting bit BUM) while the IIC bus is active (indicated by bit BB = 1). Bit AL must be cleared via software.

25.1.3.3 Operation in Slave Mode

If the on-chip IIC module shall be controlled via the IIC bus by a remote master (i.e. be a bus slave) slave mode must be selected via bit field MOD in register SYSCON. The physical channel is configured by a control word written to register BUSCON, defining the active interface pins and the used baud rate. It is recommended to have only one

SDA and SCL line active at a time when operating in slave mode. The address by which the slave module can be selected is written to register BUSCON.

The IIC module is selected by another master when it receives (after a start condition) either its own device address (stored in BUSCON) or the general call address (00_H). In this case an interrupt is generated and bit SLA in register SYSCON is set indicating the valid selection. The desired transfer mode is then selected via bit TRX (TRX = 0 for reception, TRX = 1 for transmission).

For a transmission the respective data byte is placed into the buffer RTB0 … 3 (which automatically sets bit TRX) and the acknowledge behavior is selected via bit ACKDIS.

For a reception the respective data byte is fetched from the buffer RTB0 … 3 after IRQD has been activated.

In both cases the data transfer itself is enabled by clearing bits IRQD, IRQP and IRQE which releases the SCL line.

When a stop condition is detected bit SLA is cleared.

The IIC control register CR selects the bus baud rate as well as the activation of SDA and SCL lines. So an external IIC channel can be established (baud rate and physical lines) with one single register access.

Systems that utilize several IIC channels can prepare a set of control words which configure the respective channels. By writing one of these control words to BUSCON the respective channel is selected. Different channels may use different baud rates. Also different operating modes can be selected, e.g. enabling all physical interfaces for a broadcast transmission.

25.1.4 Baud Rate Selection

In order to give the user high flexibility in selection of CPU frequency and baud rate without constraints to baud rate accuracy a flexible baud-rate generator has been implemented. It uses two different modes and an additional pre-divider. Low baud rates may be configured at high precision in mode 0 which is compatible with older versions. High baud rates may be configured precisely in mode 1.

Mode 0: Reciprocal Divider

The resulting baud rate is:

$$B_{IIC} = \frac{f_{sys}}{4 \times (BRP + 1)} \qquad BRP = \frac{f_{sys}}{4 \times B_{IIC}} - 1 \tag{25.1}$$

Mode 1: Fractional Divider

The resulting baud rate is:

$$B_{IIC} = \frac{f_{sys} \times BRP}{1024} \quad BRP = \frac{1024 \times B_{IIC}}{f_{sys}} \tag{25.2}$$

Table 25-1 IIC Bus Baud Rate Selection

BRPMOD = 0	BRP @ 100 kBaud		BRP @ 400 kBaud	
f_{sys} [MHz]	PREDIV = 00_B	PREDIV = 01_B	PREDIV = 00_B	PREDIV = 01_B
100	$F9_H$	$1E_H$	$3E_H$	07_H
50	$7C_H$	$0F_H$	$1E_H$	03_H
24	$3B_H$	06_H	$0E_H$	–
20	31_H	05_H	$0C_H$	–
16	27_H	04_H	09_H	–
12	$1D_H$	–	06_H	–
8	13_H	–	04_H	–
5	$0C_H$	–	02_H	–

BRPMOD = 1	BRP @ 100 kBaud			BRP @ 400 kBaud		
f_{sys} [MHz]	PREDIV = 00_B	PREDIV = 01_B	PREDIV = 10_B	PREDIV = 00_B	PREDIV = 01_B	PREDIV = 10_B
100	–	–	42_H	–	21_H	–
50	–	–	83_H	–	42_H	–
24	–	22_H	–	–	89_H	–
20	–	29_H	–	–	$A4_H$	–
16	–	33_H	–	–	CD_H	–
12	–	44_H	–	22_H	–	–
8	–	66_H	–	33_H	–	–
5	–	$A4_H$	–	52_H	–	–

Note: Correct functionality can only be guaranteed for 16 MHz < f_{sys} < 100 MHz.

25.1.5 Interrupts

Table 25-2 Interrupt Sources

Interrupt	Signal	SRC Register	Description
Data	IC_INT_D_O	XP0SRC	Interrupt is requested after the acknowledge bit of the last byte has been received or transmitted.
Data Error	IC_INT_D_O	XP0SRC	Interrupt is requested if a multi byte write could not be finished in slave mode because of missing acknowledge, then the data interrupt is followed by an end of transmission interrupt.
Protocol: Arbitration Lost	IIC_INT_P_O	XP1SRC	Interrupt is requested when the IIC module has tried to become master on the bus but has lost the arbitration.
Protocol: Slave Mode after Lost Arbitration	IIC_INT_P_O	XP1SRC	Interrupt is requested if multimaster mode is selected and the IIC module temporarily switches to slave mode after a lost arbitration.
Protocol: Slave Mode after Device Address	IIC_INT_P_O	XP1SRC	Interrupt is requested if multimaster mode is selected and the IIC module temporarily switches to slave mode after a lost arbitration.
Data Transmission End after Stop Condition	IIC_INT_E_O	XP2SRC	Interrupt is requested after transmission is finished by a stop condition.
Data Transmission End after RSC Condition	IIC_INT_E_O	XP2SRC	Interrupt is requested after transmission is finished by a repeated start condition (RSC).
Data Transmission End after missing Acknowledge	IIC_INT_E_O	XP2SRC	Interrupt is also requested if a transmission is stopped by a missing acknowledge.

25.1.6 Synchronization

In Mastermode, the SCL line is controlled by the IIC Module. Sent and received data is only valid if SCL is high. With SCL going down, all modules are starting to count down their low period. During the low period all connected modules are allowed to hold SCL low. As the physical bus connection is wired-AND, SCL will remain low until the device with the longest low period enters high state. Then the device with the shortest high period will pull SCL low again.

25.1.7 Programming

It is strictly recommended not to write to the IIC registers except for interrupt handling, when the IIC is working. This is indicated by the BUM bit (in master mode) and the interrupt flags. In initial mode all registers can be written. In master mode the IIC is working as long as the BUM bit is set, in slave mode the IIC is working from receiving a start condition until receiving the next stop condition. Change of transmit direction is possible only after a protocol interrupt (IRQP) or in initialization mode (MOD = 00_B).

25.1.7.1 Initialization

Before data can be sent or received data buffer size must be set in the bit field CI (only necessary if buffer greater than one byte is available). To decide if slave/master or multimaster mode is required, the MOD bits must be programmed.

25.1.7.2 Repeated Start Condition

The RSC bit must be set to one.

25.1.7.3 Start Condition

To generate a start condition the IIC must be in master mode. If the BUM bit is set, a start condition is sent and the transmission is started. The slave returns the acknowledge bit, which is indicated by the LRB bit.

25.1.7.4 Sending Data Bytes

To send bytes it is only necessary to write data bytes to the transmit buffer every time a data interrupt (IRQD) occurs.

25.1.7.5 Stop Condition

The BUM bit must be set to zero, or the STP bit must be set to one.

25.1.7.6 Receiving Data Bytes

To receive bytes it is necessary to set the TRX bit to zero. The bytes can be read after every data interrupt (IRQD). After a stop condition (protocol interrupt IRQE), the count bit field CO must be read in case of buffer size (defined in CI) is greater than one byte to decide which bytes in the receive buffer were received in the last transmission cycle.

25.2 IIC Kernel Registers

Figure 25-4 and Table 25-3 show all registers associated with the IIC Kernel.

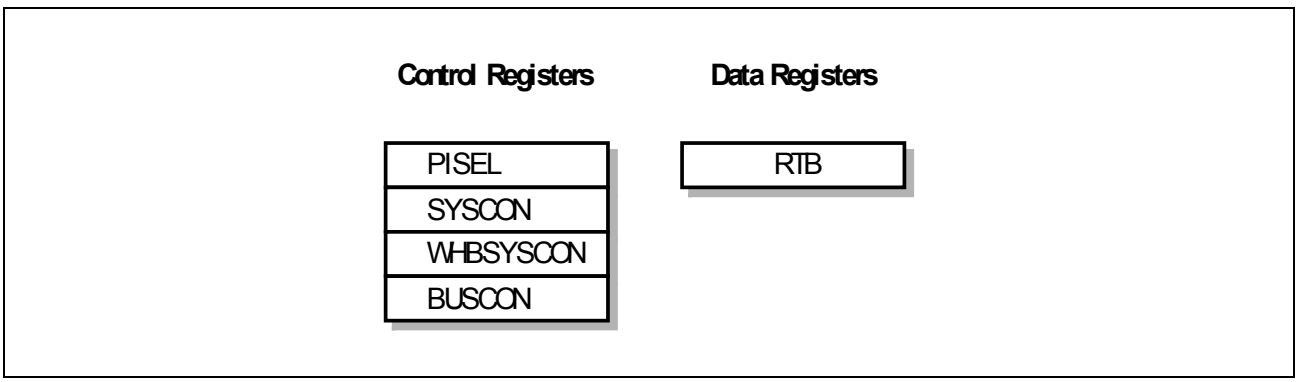


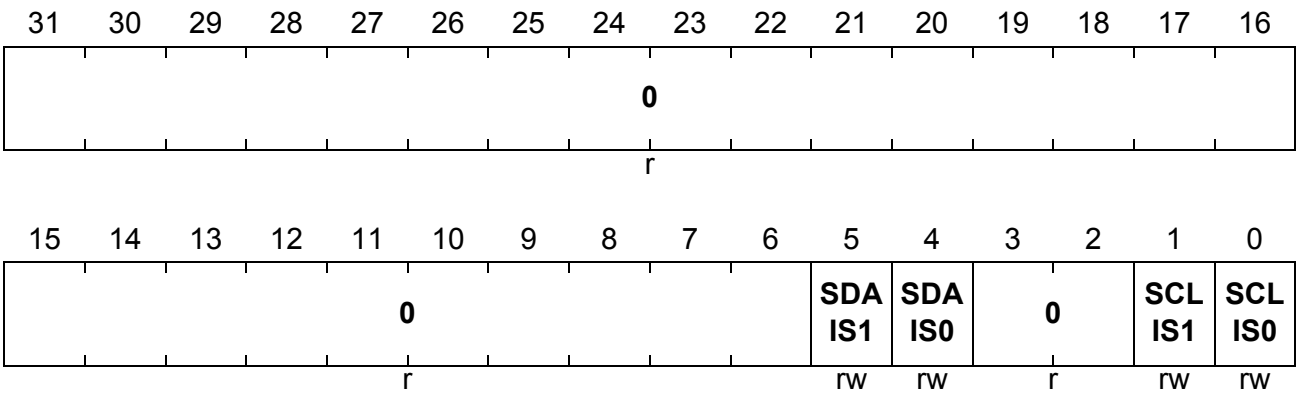
Figure 25-4 IIC Kernel Registers

Table 25-3 IIC Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
PISEL	Port Input Select Register	0004_H	Page 25-13
SYSCON	System Control Register	0010_H	Page 25-14
WHBSYSCON	Write Hardware Bits Control Register	0020_H	Page 25-20
BUSCON	Bus Control Register	0014_H	Page 25-23
RTB	Receive Transmit Buffer Register	0018_H	Page 25-25

For switching between different port input sources the IIC module provides input multiplexer, allowing selection between different input sources. This multiplexer is controlled by the PISEL register.

PISEL
Port Input Select Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
SCLISx (x = 1 … 0)	[1:0]	rw	**Select Input for Clock Signal x** 0 Port A, Default input port 1 Port B, Alternate input port
SDAISx (x = 1 … 0)	[5:4]	rw	**Select Input for Data Signal x** 0 Port A, Default input port 1 Port B, Alternate input port
0	[3:2], [31:6]	r	**Reserved;** read as 0; should be written with 0.

The operating mode of the IIC is controlled by the system control register SYSCON. This register contains control bits for mode and error check selection, and status flags for error identification. Depending on bits WMEN and RMEN, either write mirror or receive mirror is enabled. The setting and clearing of the bits SYSCON.AL, SYSCON.IRQD, SYSCON.IDQP, SYSCON.IDQe, SYSCON.RMEN, SYSCON.RSC, SYSCON.BUM, SYSCON.ACKDIS, SYSCON.TRX, SYSCON.STP and SYSCON.WMEN must be done via the special SYSCON register WHBSYSCON for the setting/clearing of those single bits.

SYSCON
System Control Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WM EN	0	0	0	RM/ CI		STP	IGE	TRX	INT	ACK DIS	BUM	MOD		RSC	M10
rwh	r	r	r	rw		rwh	rw	rwh	rw	rwh	rwh	rw		rwh	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RM EN	0	0	0	WM/ 0		CO		IRQE	IRQP	IRQ D	BB	LRB	SLA	AL	ADR
rw	r	r	r	r		rh		rwh	rwh	rwh	rh	rh	rh	rwh	rh

Field	Bits	Type	Description
ADR	0	rh	**Address** Bit ADR is set after a start condition in slave mode until the address has been received (1 byte in 7-bit address mode, 2 bytes in 10-bit address mode).
AL	1	rwh	**Arbitration Lost** Bit AL is set when the IIC module has tried to become master on the bus but has lost the arbitration. Operation is continued until the 9th clock pulse. If multimaster mode is selected the IIC module temporarily switches to slave mode after a lost arbitration. Bit IRQP is set along with bit AL. AL must be cleared via software.
SLA	2	rh	**Slave** 0 The IIC module is not selected as a slave, or the module is in master mode 1 The IIC module has been selected as a slave (device address received)
LRB	3	rh	**Last Received Bit** Bit LRB represents the last bit (i.e. the acknowledge bit) of the last transferred frame. It is automatically set to zero by a write or read access to the buffer RTB0 … 3. *Note: If LRB is high (no acknowledge) in slave mode, TRX bit is set automatically.*

Field	Bits	Type	Description
BB	4	rh	**Bus Busy** 0 The IIC bus is idle, i.e. a stop condition has occurred 1 The IIC bus is active, i.e. a start condition has occurred *Note: Bit BB is always 0 while the IIC module is disabled.*
IRQD	5	rwh	**IIC Interrupt Request Bit for Data Transfer Events**[1)] 0 No interrupt request pending 1 A data transfer event interrupt request is pending IRQD is set after the acknowledge bit of the last byte has been received or transmitted, and is cleared automatically upon a complete read or write access to the buffer(s) RTB0 … 3. New data transfers will start immediately after clearing IRQD. Do not access any register until next interrupt. If a multi byte write could not be finished in slave mode because of missing acknowledge, then the data interrupt is followed by an end of transmission interrupt. The number of bytes sent can be read from CO. The data interrupt must have higher priority than IRQE.
IRQP	6	rwh	**IIC Interrupt Request Bit for Protocol Events**[1)] 0 No interrupt request pending 1 A protocol event interrupt request is pending IRQP is set when bit SLA or bit AL is set (↙), and must be cleared via software. If the IIC has been selected by another master, the software must look up the required transmission direction by reading the received address and direction bit, stored in RTB0. The TRX-bit must be set by software correspondingly.

Field	Bits	Type	Description
IRQE	7	rwh	**IIC Interrupt Request Bit for Data Transmission End**[1)] 0 No interrupt request pending 1 A receive end event interrupt request is pending (a stop is detected) IRQE is automatically cleared upon a start condition. IRQE is not activated in init-mode. IRQE must always be deleted to continue transmission. *Note: In slave mode IRQE is set after the transmission is finished. This can also be after a stop or RSC condition. In this case the slave is not selected any more. This bit is also set, if a transmission is stopped by a missing acknowledge. In this case the bit must be cleared by software.*
CO	[10:8]	rh	**Counter of Transmitted Bytes Since Last Data Interrupt** If a multi byte transmission could not be finished because of missing acknowledge, the number of correctly transferred bytes can be read from CO. It is automatically set to zero by the correct number (defined by CI) of write/read accesses to the buffers ICRTB0 … 3. 000 No Byte 001 1 Byte 010 2 Bytes 011 3 Bytes 100 4 Bytes The number of legal bytes depends on the data buffer size (CI). Writing to this bit field does not affect its content. If WMEN is set, WM is mirrored here.
0	[14:11]	r	**Reserved;** do not use, read/write zero.

Field	Bits	Type	Description
RMEN	15	rwh	**Read Mirror Enable** 0 Read mirror is not active 1 Read mirror is active *Note: If WMEN is set RMEN cannot be set and will remain zero. If RMEN and WMEN are set simultaneously to 1, both will remain what they are. So only one of both can be set to 1.*
WM	[15:8]	wh	**Write Mirror** If WMEN is set, RTB0 may be written here. Reading WM will result zero.
M10	16	rw	**Address Mode** 0 7-bit addressing using ICA7 … 1 1 10-bit addressing using ICA9 … 0
RSC	17	rwh	**Repeated Start Condition** 0 No operation 1 Generate a repeated start condition in (multi) master mode. RSC cannot be set in slave mode *Note: RSC is cleared automatically after the repeated start condition has been sent.*
MOD	[19:18]	rw	**Basic Operating Mode** 00 IIC module is disabled and initialized (Init-Mode). Transmissions under execution will be aborted 01 Slave mode 10 Master mode 11 Multi-Master mode
BUM	20	rwh	**Busy Master** 0 Clearing bit BUM (↘) generates a stop condition immediately 1 Setting bit BUM (↗) generates a start condition in (multi)master mode *Note: Setting bit BUM (↗) while BB = 1 generates an arbitration lost situation.* *In this case BUM is cleared and bit AL is set.* *BUM cannot be set in slave mode.*

Field	Bits	Type	Description
ACKDIS	21	rwh	**Acknowledge Pulse Disable** 0 An acknowledge pulse is generated for each received frame 1 No acknowledge pulse is generated *Note: ACKDIS is automatically cleared by a stop condition.*
INT	22	rw	**Interrupt Delete Select** 0 Interrupt flag IRQD is deleted by read/write to RTB0 … 3 1 Interrupt flag IRQD is not deleted by read/ write to RTB0 … 3
TRX	23	rwh	**Transmit Select** 0 No data is transmitted to the IIC bus 1 Data is transmitted to the IIC bus *Note: TRX is set automatically when writing to the transmit buffer. It is not allowed to delete this bit in the same buscycle. It is automatically cleared after last byte as slave transmitter.*
IGE	24	rw	**Ignore IRQE** Ignore IRQE (End of transmission) interrupt. 0 The IIC is stopped at IRQE interrupt 1 The IIC ignores the IRQE interrupt *Note: If RMEN is set, RM is mirrored here.*
STP	25	rwh	**Stop Master** 0 Clearing bit STP generates no stop condition 1 Setting bit STP generates a stop condition after next transmission. BUM is set to zero. ACKDIS is set to one *Note: STP is automatically cleared by a stop condition.* *If RMEN is set, RM is mirrored here.*
CI	[27:26]	rw	**Length of the Receive/Transmit Buffer** 00 1 Byte 01 2 Bytes 10 3 Bytes 11 4 Bytes *Note: If RMEN is set, RM is mirrored here.*
0	[30:28]	r	**Reserved;** do not use, read/write zero.

Field	Bits	Type	Description
WMEN	31	rwh	**Write Mirror Enable** 0 Write mirror is not active 1 Write mirror is active *Note: If RMEN is set WMEN cannot be set and will remain zero. If WMEN and RMEN are set simultaneously to 1, both will remain what they are. So only one of both can be set to 1.*
RM	[31:24]	rh	**Read Mirror** If RMEN is set, RTB0 may be read here. Writing to RM has no effect in this mode.

1) While either IRQD, IRQP **or** IRQE is set and the IIC module is in master mode or has been selected as a slave, the IIC clock line is held low which prevents further transfers on the IIC bus.
The clock line of the IIC bus is released when IRQD, IRQE **and** IRQP are cleared. Only in this case the next IIC bus action can take place.
Interrupt request bits may be set or cleared via software, e.g. to control the IIC bus.

*Note: This register contains critical rwh bits. In case of 32-bit bus systems, these bits may not be modified directly. Please use the associated **WHBSYSCON** register instead.*

WHBSYSCON
Write Hardware Bits Control Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SETWMEN	CLRWMEN	0			SETSTP	CLRSTP	SETTRX	CLRTRX	SETACKDIS	CLRACKDIS	SETBUM	CLRBUM	0	SETRSC	CLRRSC
w	w	r			w	w	w	w	w	w	w	w	r	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETRMEN	CLRRMEN	0			SETIRQE	SETIRQP	SETIRQD	CLRIRQE	CLRIRQP	CLRIRQD	0		SETAL	CLRAL	0
w	w	r			w	w	w	w	w	w	r		w	w	r

Field	Bits	Type	Description
CLRAL	1	w	**Clear Arbitration Lost Bit** Writing 1 to this bit clears bit SYSCON.AL. Writing 0 has no effect. Reading returns 0 always.
SETAL	2	w	**Set Arbitration Lost Bit** Writing 1 to this bit sets bit SYSCON.AL. Writing 0 has no effect. Reading returns 0 always.
CLRIRQD	5	w	**Clear IIC Interrupt Request Bit for Data Transfer Events Bit** Writing 1 to this bit clears bit SYSCON.IRQD. Writing 0 has no effect. Reading returns 0 always.
CLRIRQP	6	w	**Clear IIC Interrupt Request Bit for Protocol Events Bit** Writing 1 to this bit clears bit SYSCON.IRQP. Writing 0 has no effect. Reading returns 0 always.
CLRIRQE	7	w	**Clear IIC Interrupt Request Bit for Data Transmission End Bit** Writing 1 to this bit clears bit SYSCON.IRQE. Writing 0 has no effect. Reading returns 0 always.

Field	Bits	Type	Description
SETIRQD	8	w	**Set IIC Interrupt Request Bit for Data Transfer Events Bit** Writing 1 to this bit sets bit SYSCON.IRQD. Writing 0 has no effect. Reading returns 0 always.
SETIRQP	9	w	**Set IIC Interrupt Request Bit for Protocol Events Bit** Writing 1 to this bit sets bit SYSCON.IRQP. Writing 0 has no effect. Reading returns 0 always.
SETIRQE	10	w	**Set IIC Interrupt Request Bit for Data Transmission End Bit** Writing 1 to this bit sets bit SYSCON.IRQE. Writing 0 has no effect. Reading returns 0 always.
CLRRMEN	14	w	**Clear Read Mirror Enable Bit** Writing 1 to this bit clears bit SYSCON.RMEN. Writing 0 has no effect. Reading returns 0 always.
SETRMEN	15	w	**Set Read Mirror Enable Bit** Writing 1 to this bit sets bit SYSCON.RMEN. Writing 0 has no effect. Reading returns 0 always.
CLRRSC	16	w	**Clear Repeated Start Condition Bit** Writing 1 to this bit clears bit SYSCON.RSC. Writing 0 has no effect. Reading returns 0 always.
SETRSC	17	w	**Set Repeated Start Condition Bit** Writing 1 to this bit sets bit SYSCON.RSC. Writing 0 has no effect. Reading returns 0 always.
CLRBUM	19	w	**Clear Busy Master Bit** Writing 1 to this bit clears bit SYSCON.BUM. Writing 0 has no effect. Reading returns 0 always.

Field	Bits	Type	Description
SETBUM	20	w	**Set Busy Master Bit** Writing 1 to this bit sets bit SYSCON.BUM. Writing 0 has no effect. Reading returns 0 always.
CLRACKDIS	21	w	**Clear Acknowledge Pulse Disable Bit** Writing 1 to this bit clears bit SYSCON.ACKDIS. Writing 0 has no effect. Reading returns 0 always.
SETACKDIS	22	w	**Set Acknowledge Pulse Disable Bit** Writing 1 to this bit sets bit SYSCON.ACKDIS. Writing 0 has no effect. Reading returns 0 always.
CLRTRX	23	w	**Clear Transmit Select Bit** Writing 1 to this bit clears bit SYSCON.TRX. Writing 0 has no effect. Reading returns 0 always.
SETTRX	24	w	**Set Transmit Select Bit** Writing 1 to this bit sets bit SYSCON.TRX. Writing 0 has no effect. Reading returns 0 always.
CLRSTP	25	w	**Clear Stop Master Bit** Writing 1 to this bit clears bit SYSCON.STP. Writing 0 has no effect. Reading returns 0 always.
CLRWMEN	30	w	**Set Write Mirror Enable Bit** Writing 1 to this bit sets bit SYSCON.WMEN. Writing 0 has no effect. Reading returns 0 always.
SETWMEN	31	w	**Clear Write Mirror Enable Bit** Writing 1 to this bit clears bit SYSCON.WMEN. Writing 0 has no effect. Reading returns 0 always.
0	0, [4:3], [13:11], 18, [29:27]	r	**Reserved;** read as 0; should be written with 0.

BUSCON
Bus Control Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRP MOD	PREDIV		0			ICA9 /0	ICA8 /0	ICA7..1							ICA0 /0
rw	rw		r			rw	rw	rw							rw

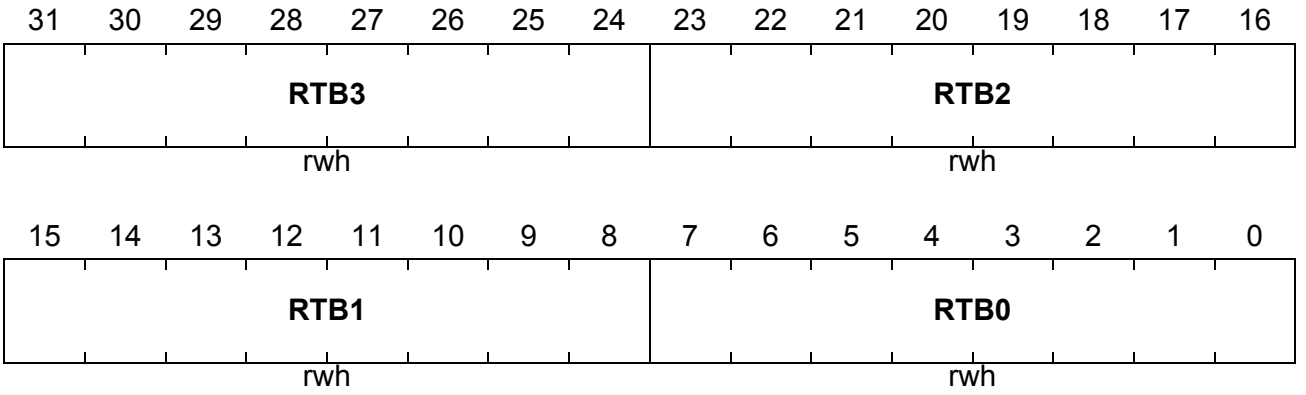
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRP								0		SCL EN1	SCL EN0	0		SDA EN1	SDA EN0
rw								r		rw	rw	r		rw	rw

Field	Bits	Type	Description
SDAENx (x = 1 … 0)	[1:0]	rw	**Enable Input for Data Pin x** These bits determine to which pins the IIC data line is connected. 0 SDA pin x is disconnected 1 SDA pin x is connected with IIC data line
SCLENx (x = 1 … 0)	[5:4]	rw	**Enable Input for Clock Pin x** These bits determine to which pins the IIC clock line is connected. 0 SCL pin x is disconnected 1 SCL pin x is connected with IIC clock line
BRP	[15:8]	rw	**Baud Rate Prescaler** Determines the baud rate for the active IIC channel(s). The prescaler may operate in two modes. Bit BRPMOD selects the actual mode. Bit field PREDIV selects an additional pre-divider. *Note: See **Table 25-1**.*
ICA0	16	rw	**Node Address Bit 0 in 10-Bit Mode** (See SYSCON bit M10) *Note: Access is only possible in 10-bit mode.*
0	16, [25:24]	r	**Reserved;** read/write 0 if in 7-bit mode
ICA7 … 1	[23:17]	rw	**Node Address in 7-Bit Mode** (ICA9, ICA8, and ICA0 disregarded)

Field	Bits	Type	Description
ICA9 … 0	[25:16]	rw	**Node Address in 10-Bit Mode** (all bits used) *Note: Access is only possible in 10-bit mode.*
PREDIV	[30:29]	rw	**Pre-divider for Baud Rate Generation** 00 Pre-divider is disabled 01 Pre-divider factor 8 is enabled 10 Pre-divider factor 64 is enabled 11 Reserved, do not use *Note: See **Table 25-1**.*
BRPMOD	31	rw	**Baud Rate Prescaler Mode** 0 Mode 0 is enabled (by default) 1 Mode 1 is enabled *Note: See **Table 25-1**.*
0	[3:2], [7:6], [28:26]	r	**Reserved;** read as 0; should be written with 0.

RTB
Receive Transmit Buffer

Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
RTBx (x = 3 … 0)	[31:0]	rwh	**Receive/Transmit Buffer**[1)] The buffers contain the data to be sent/received. The buffer size can be set in bit field CI (from 1 up to 4 bytes). RTB0 is sent/received first.

1) If bit INT is set to zero and all bytes (specified in CI) of RTB0 … 3 are read/written (dependent on bit TRX) IRQD will be cleared by hardware after completion of this access.

25.3 IIC Module Implementation

This section describes the IIC module interfaces with port connections, interrupt control, and address decoding and the requirement in TC1130.

25.3.1 Interfaces of the IIC Module

In TC1130, only two physical IIC Buses are implemented. **Figure 25-5** shows the TC1130 specific implementation details and interconnections of the IIC module. The IIC module has four I/O lines, located at Port 2. The IIC module is further supplied by a clock control, interrupt control, and address decoding logic. One DMA request can be generated by IIC module.

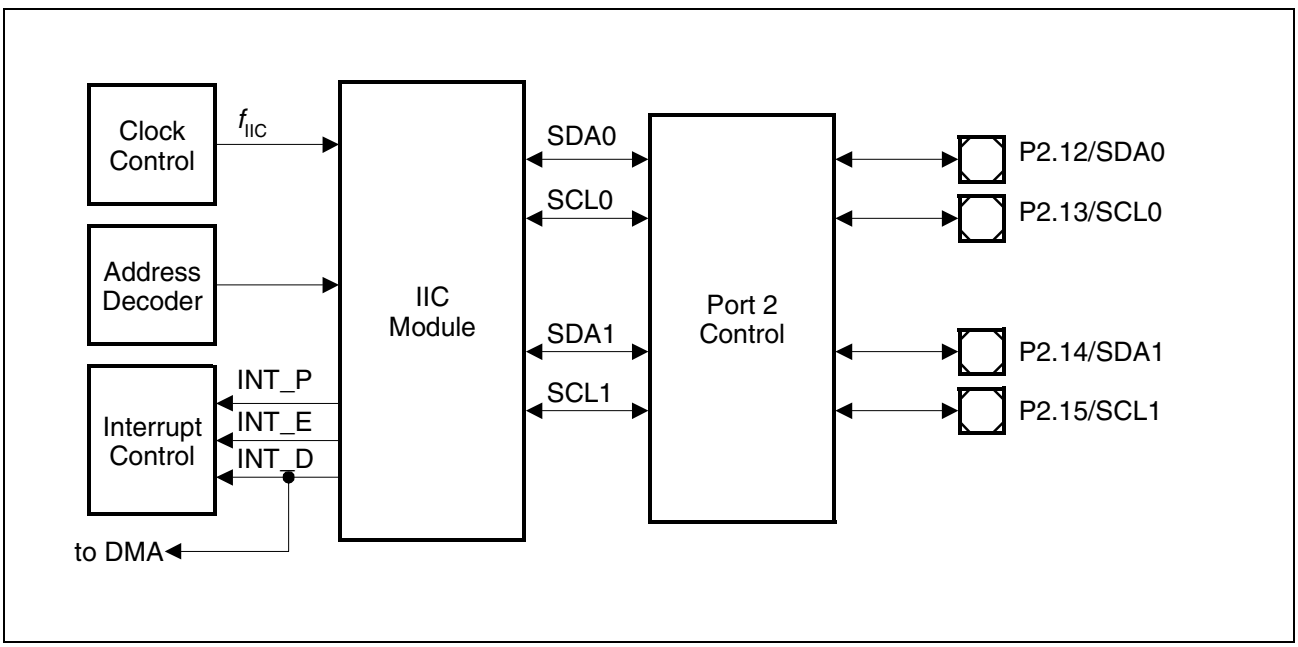


Figure 25-5 IIC Module Implementation and Interconnections

25.3.2 IIC Module Related External Registers

Figure 25-6 summarizes the module related external registers, which are required for IIC programming (see also **Figure 25-4** for the module kernel specific registers).

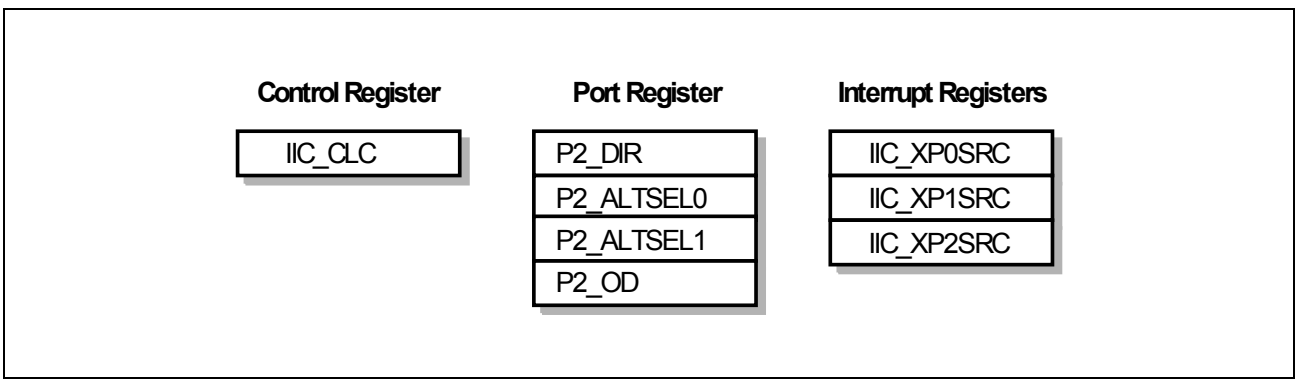
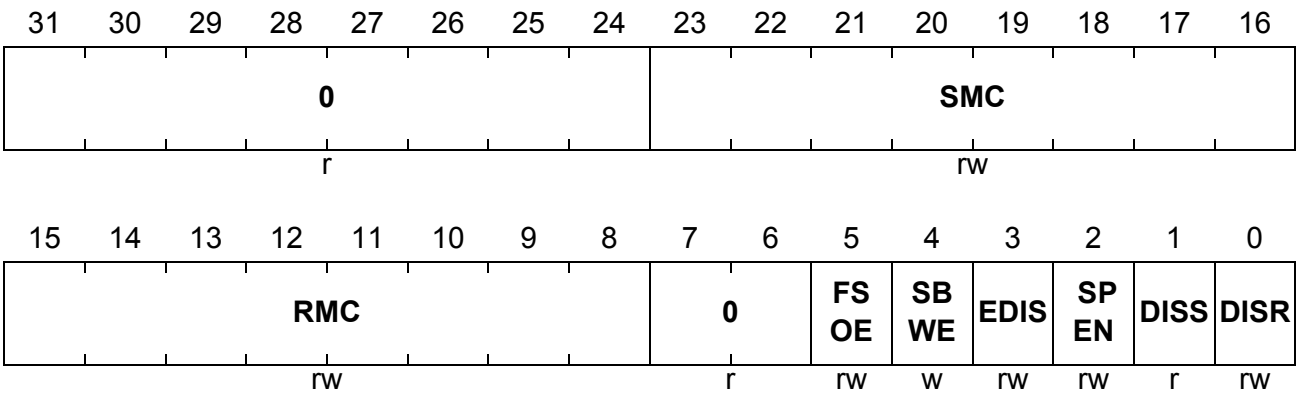


Figure 25-6 IIC Implementation Specific Special Function Registers

25.3.2.1 Clock Control Register

IIC_CLC is used to control the f_{IIC} clock signal.

IIC_CLC
IIC Clock Control Register **Reset Value: 0000 0002$_H$**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the module.
SPEN	2	rw	**Module Suspend Enable for OCDS** Used for enabling the suspend mode.
EDIS	3	rw	**External Request Disable** Used for controlling the external clock disable request.
SBWE	4	w	**Module Suspend Bit Write Enable for OCDS** Defines whether SPEN and FSOE are write protected.
FSOE	5	rw	**Fast Switch Off Enable** Used for fast clock switch off in OCDS suspend mode.
RMC	[15:8]	rw	**8-Bit Clock Divider Value in RUN Mode**
SMC	[23:16]	rw	**Clock Divider Addition for Sleep Mode** Max. 8-bit adding value. *Note: FPI Bus only.*
0	[7:6] [31:24]	r	**Reserved;** read as 0; should be written with 0.

25.3.2.2 Port Registers

The interconnections between the IIC module and the port I/O lines are controlled in the port logics. The following port control operations selections must be executed:

- Input/output function selection (DIR register)
- Alternate function selection (ALTSEL0 and ALTSEL1 registers)
- Pad driver characteristics selection for the outputs (OD register)

Input/Output Function Selection

The port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as port direction (input/output), open-drain, and alternate output selections. The I/O lines for the IIC modules are controlled by the port input/output control registers of Port 2.

Table 25-4 shows how bits and bit fields must be programmed for the required I/O functionality of the IIC I/O lines.

Table 25-4 IIC I/O Control Selection and Setup

Module	Port Lines	Input/Output Control Register Bits	I/O
IIC	P2.12/SDA0	P2_DIR.P12 = 0_B	Input
		P2_DIR.P12 = 1_B	Output
		P2_ALTSEL0.P12 = 1_B	
		P2_ALTSEL1.P12 = 0_B	
	P2.13/SCL0	P2_DIR.P13 = 0_B	Input
		P2_DIR.P13 = 1_B	Output
		P2_ALTSEL0.P13 = 1_B	
		P2_ALTSEL1.P13 = 0_B	
	P2.14/SDA1	P2_DIR.P14 = 0_B	Input
		P2_DIR.P14 = 1_B	Output
		P2_ALTSEL0.P14 = 1_B	
		P2_ALTSEL1.P14 = 0_B	
	P2.15/SCL1	P2_DIR.P15 = 0_B	Input
		P2_DIR.P15 = 1_B	Output
		P2_ALTSEL0.P15 = 1_B	
		P2_ALTSEL1.P15 = 0_B	

P2_DIR
Port 2 Direction Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 12-15)**	n	rw	**Port 2 Pin 12-15 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for IIC I/O port control.

P2_ALTSELn (n = 0, 1)
Port 2 Alternate Select Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 25-5 Function of the Bits P2_ALTSEL0.Pn and P2_ALTSEL1.Pn (n = 12-15)[1)]

P2_ALTSEL0.Pn	P2_ALTSEL1.Pn	Function
1	0	Alternate Select 1

1) Shaded bits and bit field are don't care for IIC I/O port control.

P2_OD
Port 2 Open Drain Control Register

Reset Value: 0000 F000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

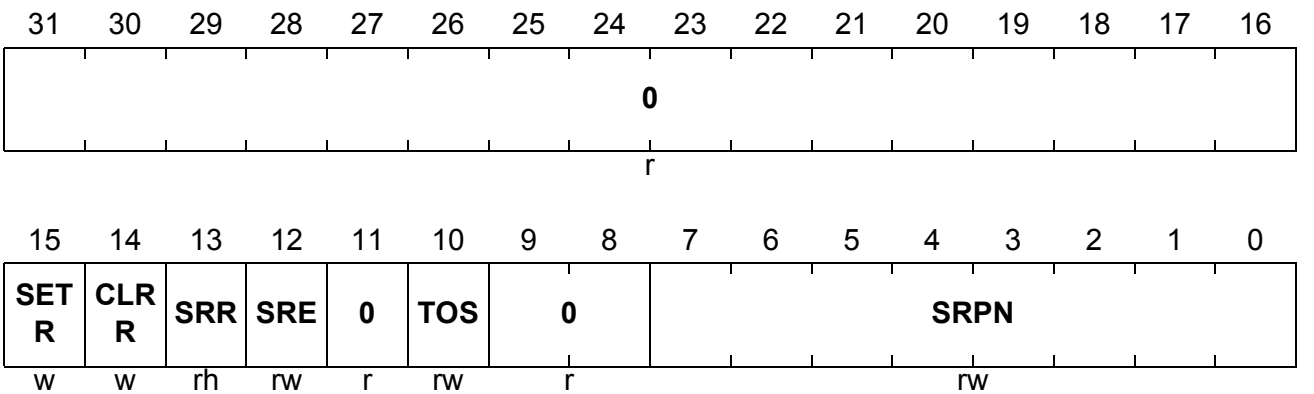
Field	Bits	Type	Description
Pn **(n = 12-15)**	n	rw	**Port 2 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for IIC I/O port control.

25.3.2.3 Service Request Control Registers

Each of the eight interrupts of the IIC module are controlled by its own service request control registers.

IIC_XP0SRC
IIC Service Request Control Register 0
IIC_XP1SRC
IIC Service Request Control Register 1
IIC_XP2SRC
IIC Service Request Control Register 2 **Reset Values: 0000 0000$_H$**



Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number**
TOS	10	rw	**Type of Service Control**
SRE	12	rw	**Service Request Enable**
SRR	13	rh	**Service Request Flag**
CLRR	14	w	**Request Clear Bit**
SETR	15	w	**Request Set Bit**
0	[9:8], 11, [31:16]	r	**Reserved;** read as 0; should be written with 0.

For proper operation of an IIC function controlled by an interrupt service routine, the following conditions should be checked:

- An interrupt request can only be serviced if the respective Service Request Enable Bit (IIC_XPxSRC.SRE) is set to 1.
- The exact source of an interrupt request should be identified by analyzing the system Control register SYSCON and WHBSYSCON.

- The Service Request Priority Number bit field SRPN defines the sequence for the CPU arbitration in case of simultaneously set Interrupt Service Request Flags. That requires careful estimation of the IIC service request priorities depending on the real time characteristic of higher prioritized interrupt sources, the CPU load, and the timing constraints to be matched by an IIC interrupt service routine.

Note: Further details on interrupt handling and processing are described in the "Interrupt System" chapter of the TC1130 System Units User's Manual.

25.3.3 DMA Requests

The DMA request lines of the IIC modules become active whenever its related interrupt line is activated. The DMA request lines are connected to the DMA controller as shown in **Table 25-6**.

Table 25-6 DMA Request Lines of IIC

Module	Related IIC Interrupt	DMA Request Line	Description
IIC	INT_D	IIC_INTD	IIC Data Interrupt DMA Request

25.3.4 IIC Register Address Range

In the TC1130, the registers of the IIC module are located within the following address range:

- Module Base Address = F010 0600_H
  Module End Address = F010 $06FF_H$
- Absolute Register Address = Module Base Address + Offset Address
  (offset addresses see **Table 25-3**)

Note: The complete and detailed address map of the IIC module is described in the chapter "Register Overview" of the TC1130 System Units User's Manual.

26 USB

This chapter describes the USB Module of the TC1130. It contains the following sections:

- Functional description of the USB Kernel (see **Section 26.1**)
- Register descriptions of all USB Kernel specific registers (see **Section 26.2**)
- TC1130 implementation specific details and registers of the USB (port connections and control, interrupt control, address decoding, clock control, see **Section 26.3**)

*Note: The USB kernel register names described in **Section 26.2** will be referenced in other parts of the TC1130 User's Manual with the module name prefix "USB_".*

26.1 USB Kernel Description

The USB is the answer to connectivity for the PC architecture. It is a fast, bidirectional, isochronous, low-cost, dynamically attachable, four wire serial interface that is consistent with the requirements of the PC platform.

26.1.1 Introduction

The USB module in the TC1130 supports USB1.1 and cannot function as host. It has the on-chip transceiver for transaction.

Features

- USB1.1 Device Standard Interface
- On-chip transceiver
- Differential I/O allow cable length up to 5 m without additional hardware at target's end
- Complete USB implementation (including pads)
- Hot attach
- USB1.1 full speed device
- USB protocol handling in hardware
- Clock and data recovery from USB
- Bit stripping and bit stuffing functions
- CRC5 checking, CRC16 generation and checking
- Serial to parallel data conversion
- Maintenance of data synchronization bits (DATA0/DATA1 Toggle Bits)
- Supports multiple configurations, interfaces and alternate settings
- Programmable number of endpoints
- User configurable endpoint information
- Flexible intermediate buffering of transmission data
- Powerful data handling capability, FIFO-support
- Scalable endpoint buffer size
- Back-to-back transfers fully supported by module automatism
- Multi packet transfer without CPU load
- Handles data transfer with minimum CPU load
- Auto increment and single address modes selectable for easy data access
- Powerful interrupt generation
- Meets suspend power consumption restrictions in Power Down Mode
- Remote wake-up from USB bus activity
- Explicit support of setup information
- Enhanced status monitoring

Hardware Restrictions / Configurations in this Implementation

- Get_Descriptor command is handled by software, not automatically
- Set_Descriptor is not supported
- The module is hard configured to be a “Not Self Powered” device
- Sync_Requests from the Device are not supported
- 2 Kbytes buffer size
- 1 Configuration (CF1)
  - CF1 with 4 interfaces
  - IF0 with 1 Alternate Setting AS0
  - IF1 with 2 Alternate Setting AS0 and AS1
  - IF2 with 3 Alternate Settings AS0, AS1, and AS2
  - IF3 with 3 Alternate SettingsAS0, AS1, and AS2
- Number of Endpoints: 11

Note: The programmer may associate any Endpoints with any existing Interface and Alternate Setting as long as this meets the above said requirements.

Table 26-1 Glossary

Term	Definition
EOT	End of Transfer
SE0	Single ended Zero (see USB Specification V1.1)
SOF	Start of Frame (see USB Specification V1.1)
USB	Universal Serial Bus

26.1.2 General Operation

The USB handles all transactions between the serial USB bus and the internal (parallel) bus of the microcontroller. The USB module includes several units, which are required to support data handling with the USB bus: the on-chip USB transceiver (optionally), the flexible USB buffer block with a 32-bit wide RAM, the buffer control unit with sub modules for USB and CPU memory access control, the UDC_IF device interface for USB protocol handling, the microcontroller interface unit (MCU) with the USB specific special function registers and the interrupt generation unit. A clock generation unit provides the clock signal for the USB module for full speed and low speed USB operation. **Figure 26-1** shows the block diagram of the functional units of the USB module with their interfaces.

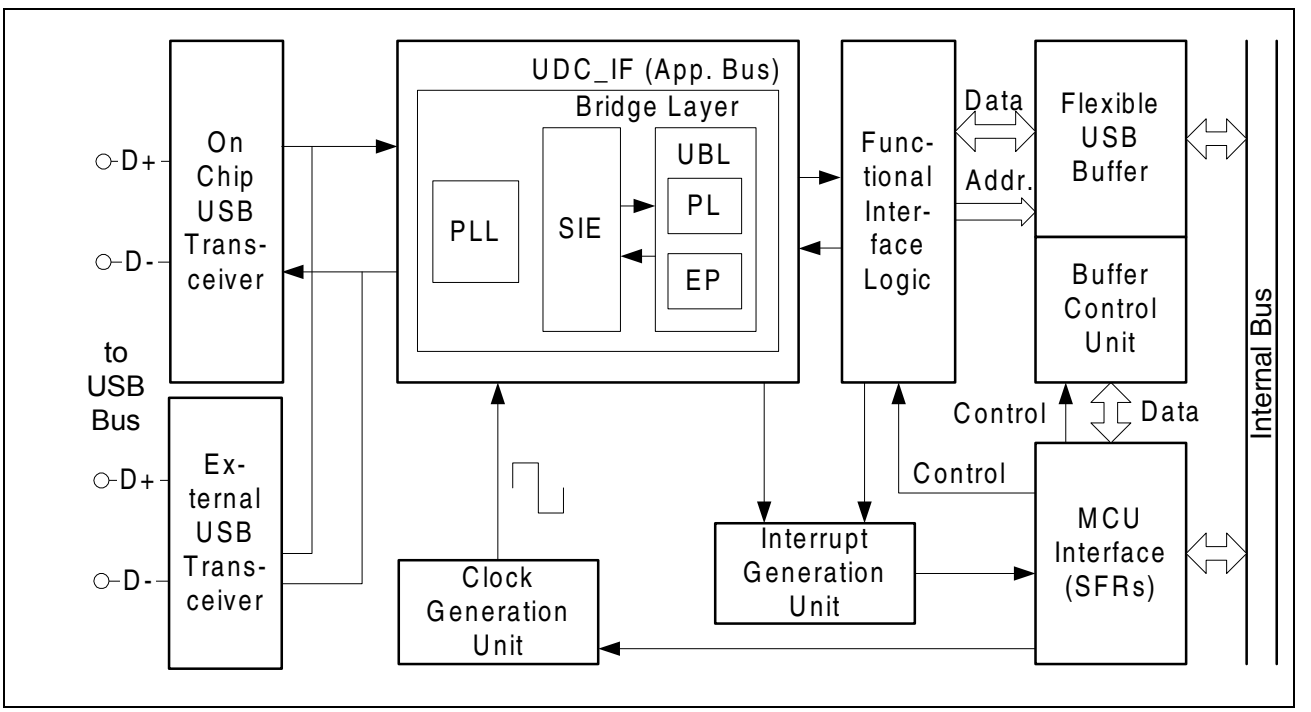


Figure 26-1 USB Block Diagram

Table 26-2 USB Signals and Interfaces

Name	Type	Description
USBCLK	I	USB input clock
VPI	I	USB D+ CMOS level mirror of differential signal
VMI	I	USB D- CMOS level mirror of differential signal
VPO	O	USB D+ CMOS level output
VMO	O	USB D- CMOS level output
RCVI	I	USB data input, combinatorical result of D+, D-
USBOE	O	Direction select Transmit/Receive

Note: The above signals are for the digital interface of the external transceiver. Only D+, D- pins are sufficient if the on-chip transceiver is used.

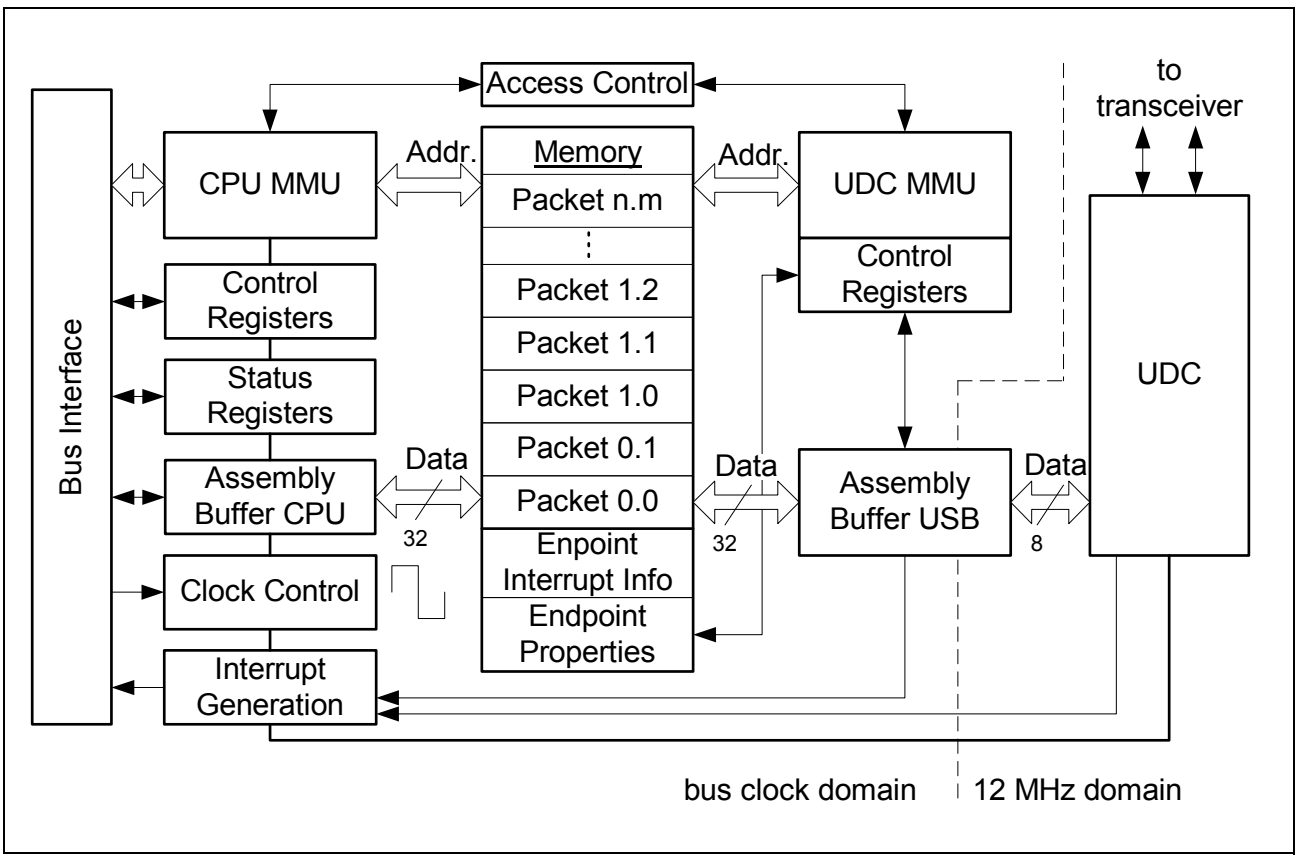


Figure 26-2 USB Detailed Functional Block Diagram

26.1.2.1 Memory

The Memory consists of one scalable single ported RAM provided by the USB kernel. It is used by the CPU and the USB concurrently. The data buffers for the Endpoints are located here as well as the Endpoint Properties, Endpoint specific pointers and Endpoint specific interrupt information.

The Endpoint Properties are written by the CPU during initialization phase to a dedicated Memory area called Endpoint Properties Area. They describe the behavior of each Endpoint according to USB specification. The USB fetches the current Endpoint Properties and provides them to the UDC automatically, if the UDC selects one Endpoint for communication.

By setting Endpoint specific pointers, the CPU determines the location and size of Buffer Area used by the referring Endpoint in the Memory. The USB uses and updates this information during operation automatically.

The Memory is also used to store and transfer interrupt information that need not to be held in registers.

To minimize the probability of conflicts of simultaneous CPU and USB accessing, special means are taken to collect smaller accesses until one 32-bit access is necessary since the memory is 32-bit wide.

26.1.2.2 Access Control

The Access Control Unit avoids access conflicts on the address and data paths of the Memory.

If the CPU and the USB try simultaneously to access the Memory the USB does the arbitration.

If an access to the Memory is already pending the requesting participant must wait until this access is finished. After completion of the pending access the waiting participant normally gets the next access to the Memory.

This simple scheme is sufficient to handle the accesses of the CPU and the USB kernel. USB internal accesses to the Memory are always 32-bit wide. This reduces the number of accesses to the Memory if data is provided externally only 8- or 16-bit wide.

The USB kernel stores data in Memory via the Assembly Buffer. The UDC MMU has the highest priority for the memory access.

26.1.2.3 Assembly Buffers

There are two Assembly Buffers implemented in the USB. They provide a 32-bit wide interface to the Memory. The CPU Assembly Buffer provides a variable access width, i.e. 8 bit, 16 bit and 32 bit to the CPU bus. The UDC Assembly Buffer provides an 8-bit wide interface required by the UDC.

The CPU Assembly Buffer reduces the number of accesses to the Memory if the CPU access is smaller than 32 bit and thereby reduces the probability of access conflicts between CPU and USB.

As the UDC data path is only 8 bit wide, the Assembly Buffer on this side is mandatory.

An Assembly Buffer consists of two 32-bit wide data registers, a multiplexer and a state machine controlling this block.

When the CPU or the UDC performs 8-bit writes, the Assembly Buffer writes the bytes to its current register in consecutive order. If a double word is written, the state machine switches its internal registers automatically. While the CPU or the UDC writes to the second internal register and the data in the first register is transferred to the Memory. Two registers are needed here for buffering as the Assembly Buffer may have to wait until the Access Control unit allows writing.

This mechanism is the same for read access to the Memory. Here data is fetched 32-bit wide from the Memory and delivered 32-, 16- or 8-bit wide to the requesting device.

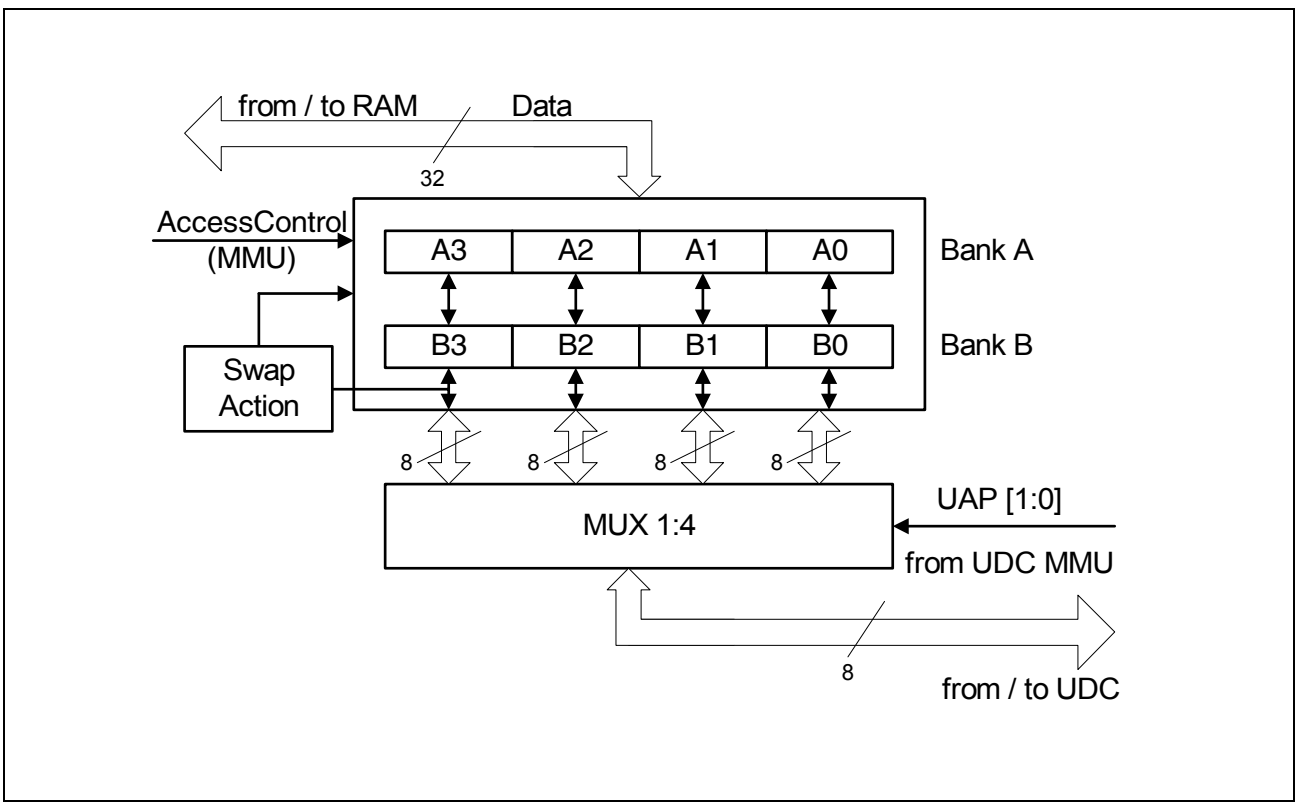


Figure 26-3 UDC Assembly Buffer

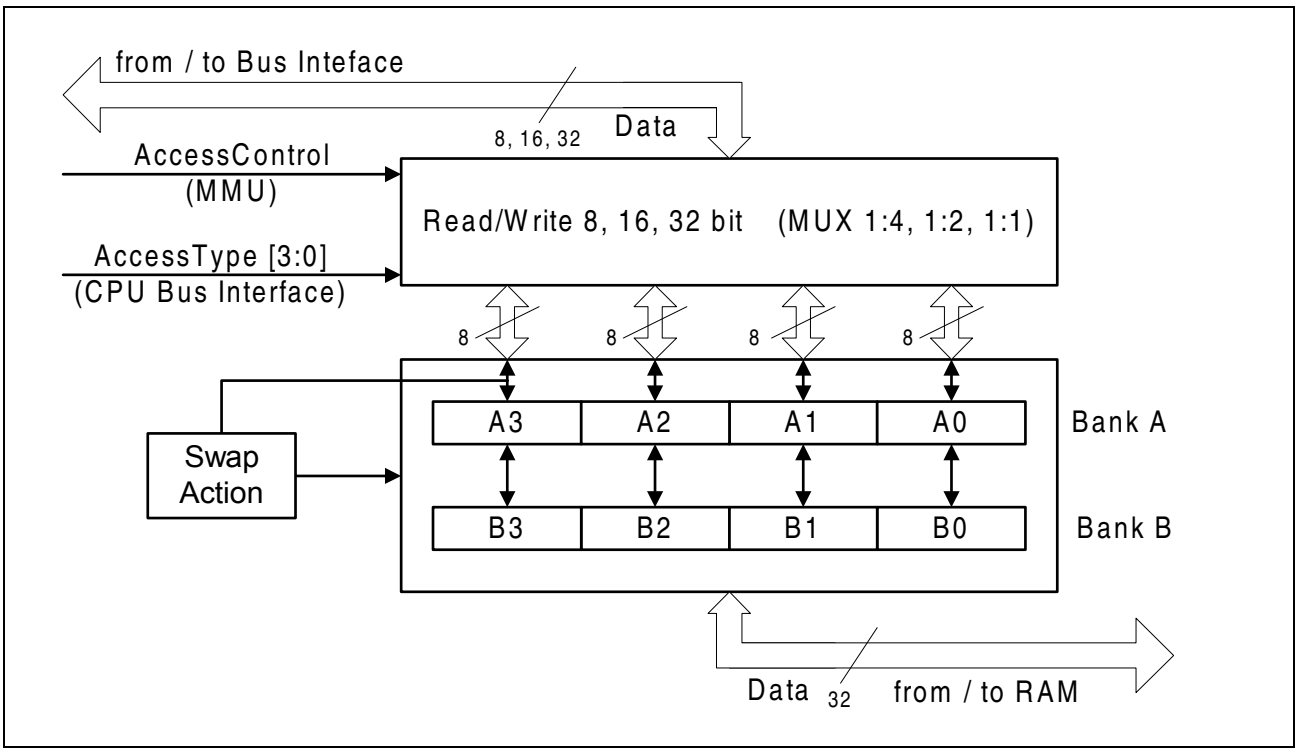


Figure 26-4 CPU Assembly Buffer

26.1.2.4 MMUs of USB

The MMUs of USB control the buffer handling of all Endpoint buffers within the Memory. The MMUs of USB consist of CPU MMU and UDC MMU. They provide pointer registers for the base address of an Endpoint buffer (Bit EPUP.BSP), the top address (the buffer length is EPUP.PSZ × EPCP.BSZ), and the current read and write pointers. All pointers for all Endpoints are set by the CPU in the Endpoint Properties Area in the Memory during initialization and can be updated during runtime.

If data for an Endpoint is addressed, both CPU MMU and UDC MMU load the corresponding set of pointers and provide the incrementation of the current pointer as well as the roll over if the top of the Endpoint buffer is reached.

The UDC MMU will write back the current pointer to the Endpoint Properties Area after successful transmission of the whole packet. If the UDC signals a transmission error, the current pointer is not written back thus, the Memory location of the start of the packet to be retransmitted is retained in the Memory.

The CPU MMU may be operated in FIFO mode (optionally). In this mode, it is possible to read and write the USB data from a single address. In addition, the current pointer may be accessed by the CPU for control functions. A simple DMA controller may be used to transfer data to or from the USB while the target address is automatically incremented/ decremented by the DMA.

The MMUs of USB provide the necessary interrupts and status flags for handling the Endpoint buffers, packets and transfers.

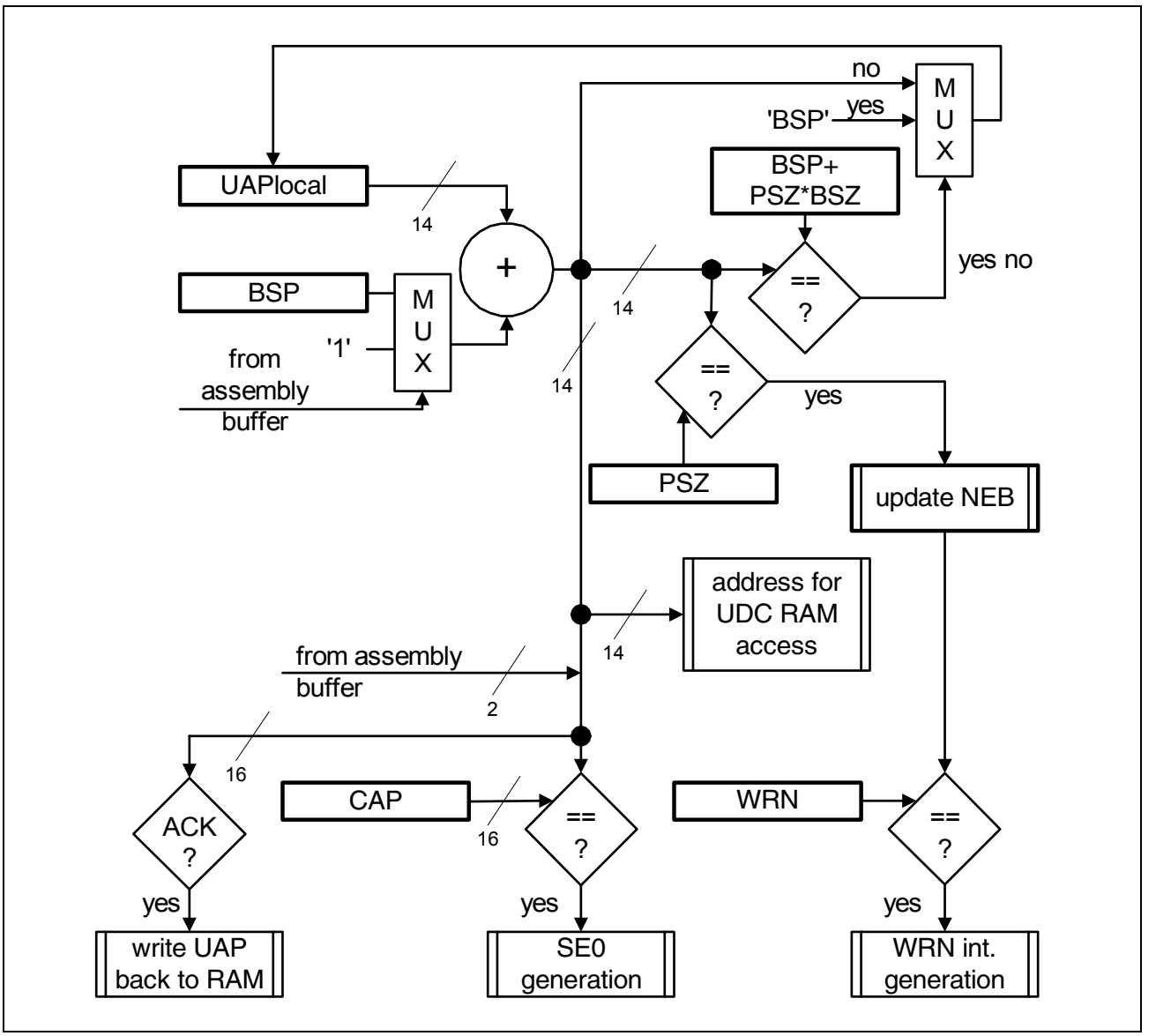


Figure 26-5 UDC MMU

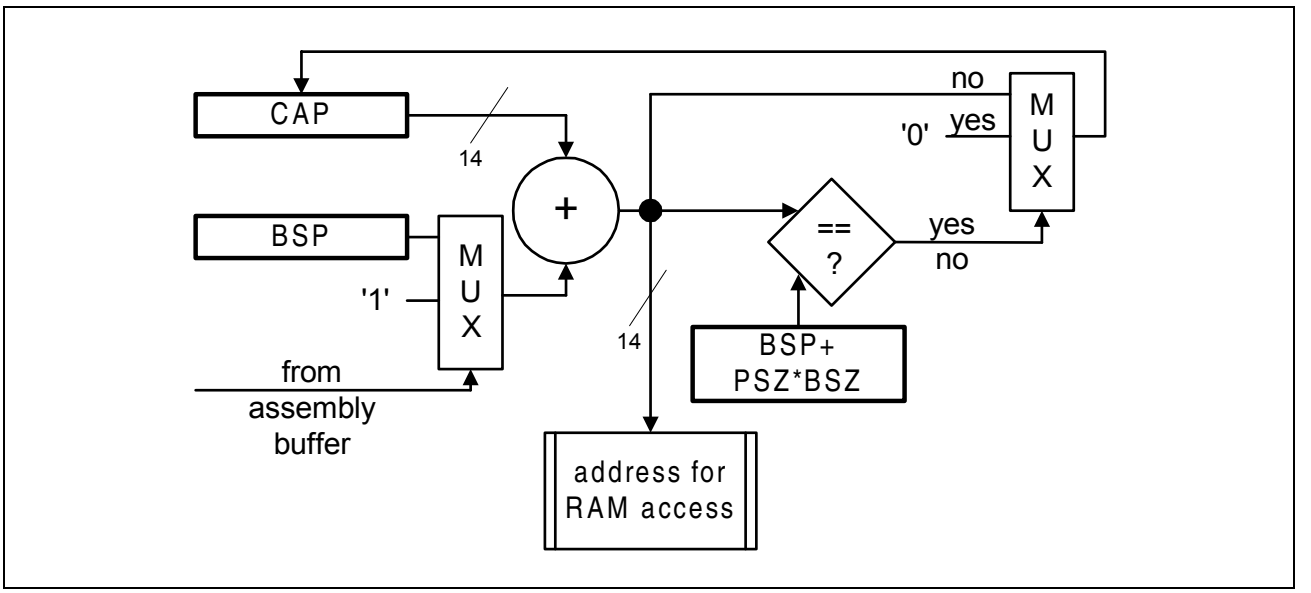


Figure 26-6 CPU MMU

The automatic FIFO support is shown in **Figure 26-7**.

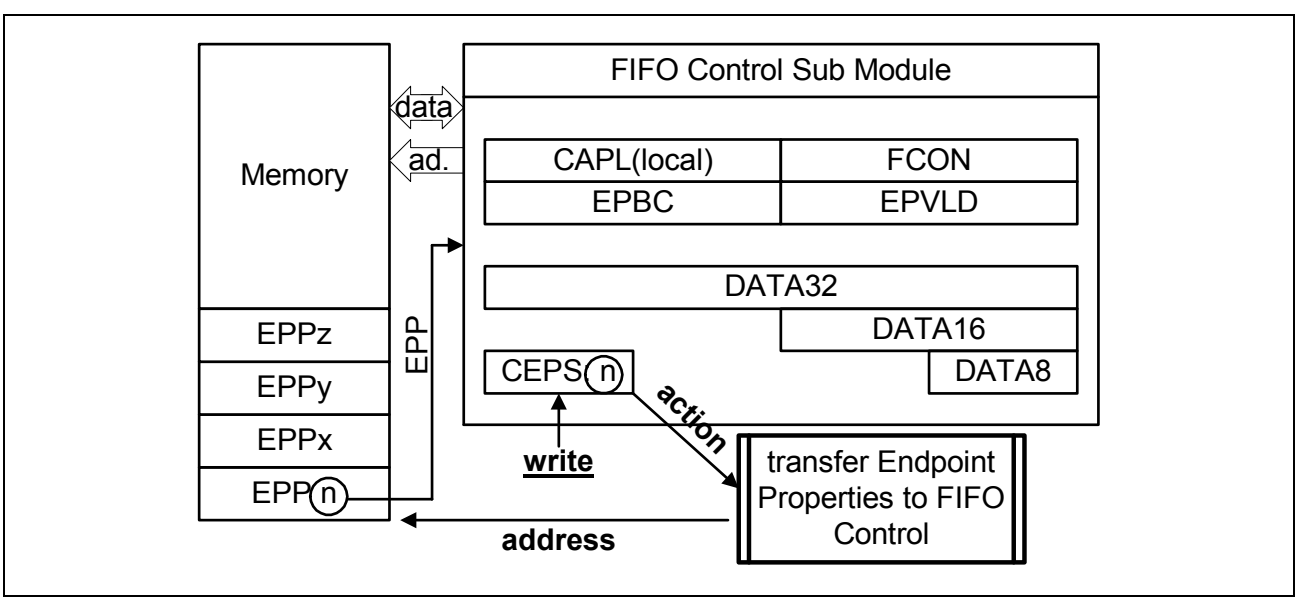


Figure 26-7 Automatic FIFO Support

26.1.2.5 Transfer Modes

USB data transfers take place between host software and a particular Endpoint on a USB device. A given USB device may support multiple data transfer Endpoints. The USB host treats communications with any Endpoint of a USB device independently from any other Endpoint. Such associations between the host software and a USB device Endpoint are called pipes. As an example, a given USB device could have an Endpoint supporting a pipe for transporting data to the USB device and another Endpoint supporting a pipe for transporting data from the USB device.

The USB architecture comprehends four basic types of data transfers.

Table 26-3 USB Transfer Modes

Mode	Function
Control	Control data are used to configure devices, data transmission is lossless. Control pipes are bidirectional, data transfer is possible in both directions via one pipe. Endpoint 0 is always configured as control Endpoint with a maximum buffer length of 8 bytes.
Isochronous (full speed mode only)	Isochronous data are continuous and real-time in creation and consumption, such as voice data. In this case, real-time is defined from frame to frame. Isochronous data transfer has the highest priority, but is not always lossless. Isochronous pipes are always unidirectional, so one Endpoint can be associated to an IN pipe or an OUT pipe. The TC1130 supports up to 1023 bytes.
Interrupt	Interrupt data are a small amount of data, which are transferred to the host every n frames, with n being programmable by the host. Data delivery is lossless. Interrupt pipes are always unidirectional IN pipes, the maximum data packet length is limited to 64 bytes.
Bulk (full speed mode only)	Bulk data can be a larger amount of data, which can be split by the host in several data packets within one frame. Data delivery is lossless. Bulk pipes are always unidirectional, so one Endpoint can be associated to an IN pipe or an OUT pipe. The maximum data packet length is limited to 64 bytes.

26.1.2.6 Initialization of USB

The USB must be functionally initialized by writing 6 configuration bytes, which contain appropriate device information.

A well defined procedure must be executed for switching on the clock of the USB module:

- The 48 MHz clock for USB must be switched on in the system
- If the clock is provided by a PLL, it is necessary to wait for the PLL to be locked
- DISR bit of the CLC register must be cleared

The switch-on procedure after a hardware reset assures proper operation of the USB clock system. When the USB clock system is switched on, a software initialization procedure must follow. This procedure must execute the following steps:

- Setting bit DCR.SWRS starts the software reset operation for the complete USB module. For a full speed device, bit DCR.SPEEDS must be set together with DCR.SWRS in the same instruction (write protection of DCR.SPEEDS bit).
- When the software reset is finished, bit DSR.SWR is cleared by hardware and bit DSR.DINIT is set automatically to indicate to the CPU the start of the initialization sequence.
- The USB module must be functionally initialized by the CPU by writing 6 configuration bytes for each Endpoint to the memory. Thereafter, bit DCR.EOID must be set by software.

Figure 26-8 shows the 6-byte configuration block, which must be transmitted for each Endpoint by the CPU to the USB module via memory access. **Table 26-4** shows the bit field definition.

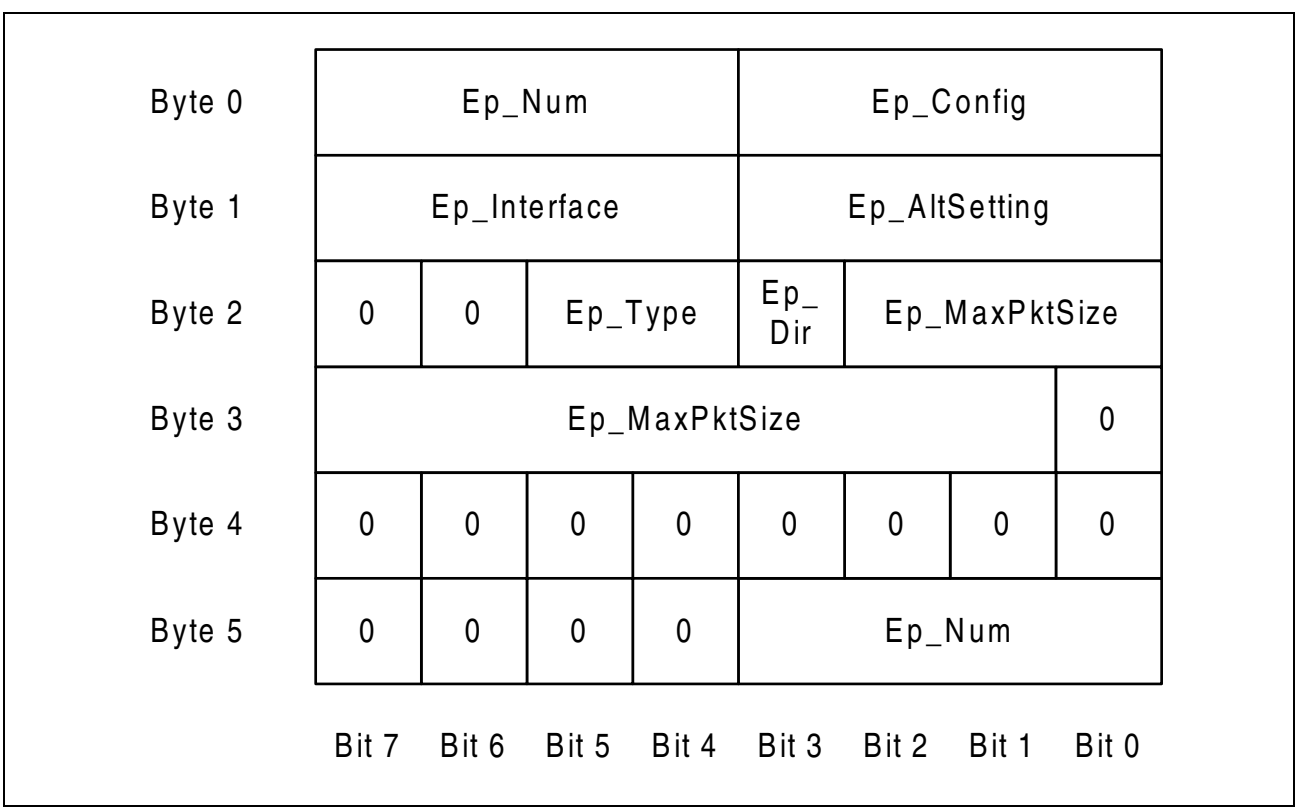


Figure 26-8 USB Configuration Block

The six byte USB configuration block must be written to the memory in sequential order (byte 0 to byte 5) from the CPU to the USB module memory (starting at memory address 0) for each Endpoint beginning with Endpoint 0, followed by the USB configuration block for Endpoint 1 and so on up to the USB configuration block for Endpoint 10. The CPU may perform 8-, 16- or 32-bit accesses. EP_Num is set for all Endpoints (1 Configuration[1)] with 4 Interfaces in up to 4 Alternate Settings). See **"Hardware**

1) Note that basically an additional Configuration "0" is present automatically after reset, containing EP0 only.

Restrictions / Configurations in this Implementation" on Page 26-3. After this action, bit DINIT is reset by hardware, and the software reset and initialization sequence are finished.

Table 26-4 Bit Field Definition of USB Configuration Block

Bit Field Name	Definition
EP_Num	This 4-bit field specifies the number of the Endpoint (0-10) for which the current configuration byte block is valid. This 4-bit field must be referenced in byte 0 and byte 5 of a configuration byte block.
EP_Config	This 4-bit field specifies the configuration number (0-1) to which this Endpoint is configured to.
EP_Interface	This 4-bit field specifies the number of the interface (0-3) to which this Endpoint is configured to.
EP_AltSetting	This 4-bit field selects the alternate setting of the corresponding interface (EP_Interface) to which this Endpoint is configured to.
EP_Type	This 2-bit field defines the type of the Endpoint: 00 Control Endpoint 01 Isochronous Endpoint 10 Bulk Endpoint 11 Interrupt Endpoint Endpoint 0 must be setup for control Endpoint.
EP_Dir	This bit defines the direction of the Endpoint: 0 Out (packets to be transferred from Host to CPU) 1 In (packets to be transferred from CPU to Host)
EP_MaxPktSize	This 10-bit field defines the maximum packet size to be transferred to this Endpoint within the range from 0 up to 1023 bytes. The configuration of EPPackSize must be in harmony with the USB specification.

26.1.2.7 USB Device Framework

Enumeration Process

The bus enumeration process consists of an interrogation sequence through which the USB host acquires information from the connected device, gives it an unique address, and assigns it a configuration value. In the simplest form, the process takes four steps:

Step 1: The host issues a Get_Descriptor command to the device through the default address using the control pipe. The device then provides information about itself, such as device class, vendor id, maximum packet size for Endpoint 0, etc.

Step 2: The host sends a unique address in a data packet to the device using the Set_Address command. The device, under software control, gets the address through Endpoint 0 and stores it.

Step 3: The host requests and reads the device configuration descriptor using the Get_Configuration command. The device responds with information about the number of interfaces and Endpoints, Endpoint transfer type, packet size and direction, maximum power requirements, power source, etc.

Step 4: The last step of the enumeration process is handled using the Set_Configuration command through which the host assigns a configuration value to the device.

After the enumeration process is complete, the device is configured and ready for USB data transmit and receive transactions.

26.1.2.8 Control Transfers

A control transfer consists of at least two and perhaps three stages. This chapter gives a short description of these stages of a control transfer and the associated control and status bits.

Setup Stage

A control transfer always begins with a setup stage that transfers information to a target device, defining the type of request being made to the USB device. The standard commands except the Set_Descriptor, Get_Descriptor and Synch_Frame commands are handled by the USB module automatically without CPU interaction. If the command is not handled by the USB module automatically, a setup interrupt (bit SUI) indicates the end of a setup phase. Additionally, the status bit EPV0 is reset.

Data Stage

This stage occurs only for requests that require data transfers. The direction of this data stage is always predicted to be from Host to Device (bit DIR is automatically cleared after the setup stage occurred). The first data packet may immediately be sent from the Host to the control Endpoint if EPV0 is set.

The direction of the next transfer can also be predicted to be a USB read access (DIR = 1). The data packet to be transferred from the CPU to the Host is setup before the next USB access occurs. Therefore, the bit EPV0 must be set, to be able to transfer the data packet within the first USB read access. Status bits FVLD/NEB is set under hardware control to indicate valid data to be read by the CPU in case of a USB write access, or data to be written by the CPU in case of a USB read access.

Status Stage

The status stage always occurs to report the result of the requested operation. A status stage is initiated by the Host although EPVAL0 is 0 indicated by a status interrupt (STI). Bit EPV0 must be set under software control to enable the acknowledge of the status stage.

Note: If the CPU detects a corrupted control transfer (Endpoint 0), bit STALL0 should be set by software in order to indicate an error condition which cannot be recovered by the USB device itself.

Standard Device Requests

Only the Set_Descriptor, Get_Descriptor and Synch_Frame requests require intervention from the CPU. The Set_Configuration and Set_Interface standard device requests are handled by the UDC core and they are transparent to firmware. Optionally more requests can be handled by the CPU if required. Since the device supports multiple device configurations, interfaces and alternate settings, a separate register CNFR informs the CPU which configuration, interface and alternate setting is active.

Standard Device Commands

The Set_Configuration and Set_Interface standard device commands are handled by the UDC and are transparent to the firmware.

26.1.2.9 USB Endpoint Buffer Organization

Figure 26-9 shows the Endpoint buffer organization with the maximum number of Endpoints and the maximum Memory size.

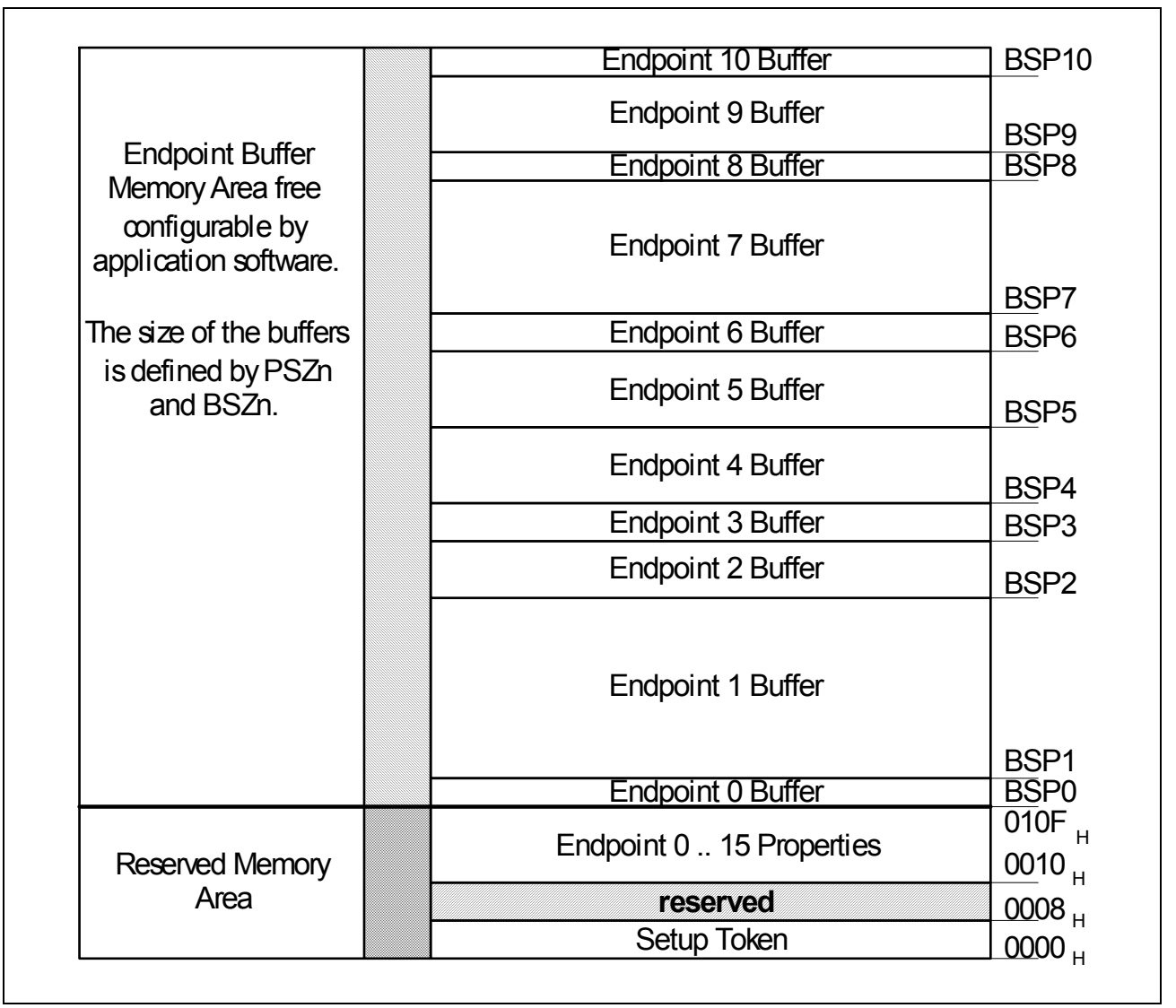


Figure 26-9 Memory Organization

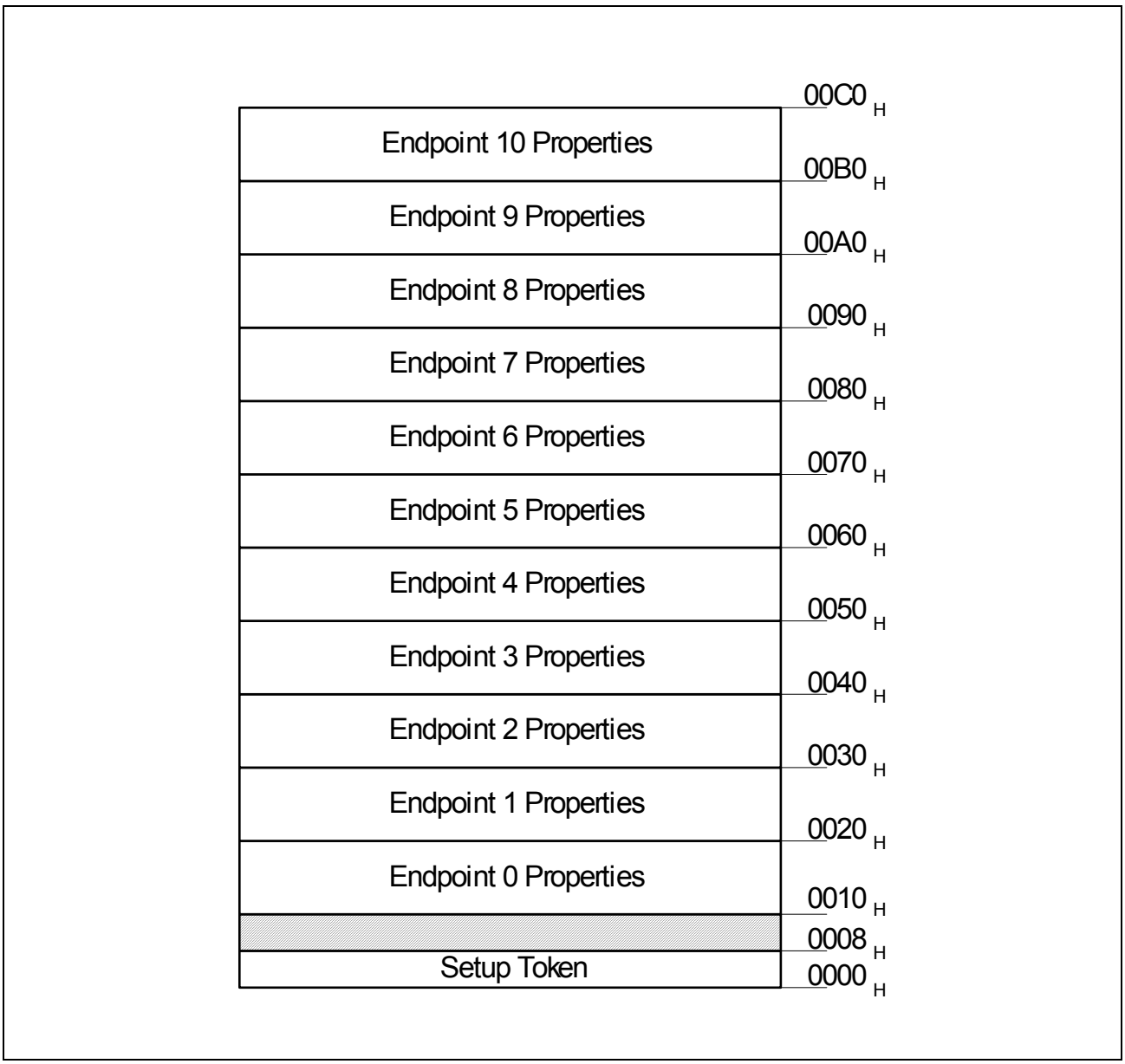


Figure 26-10 Endpoint Properties Memory Block

Note: Reduced reserved area is needed if less Endpoints are used.
Unused memory space can be used for data buffering.

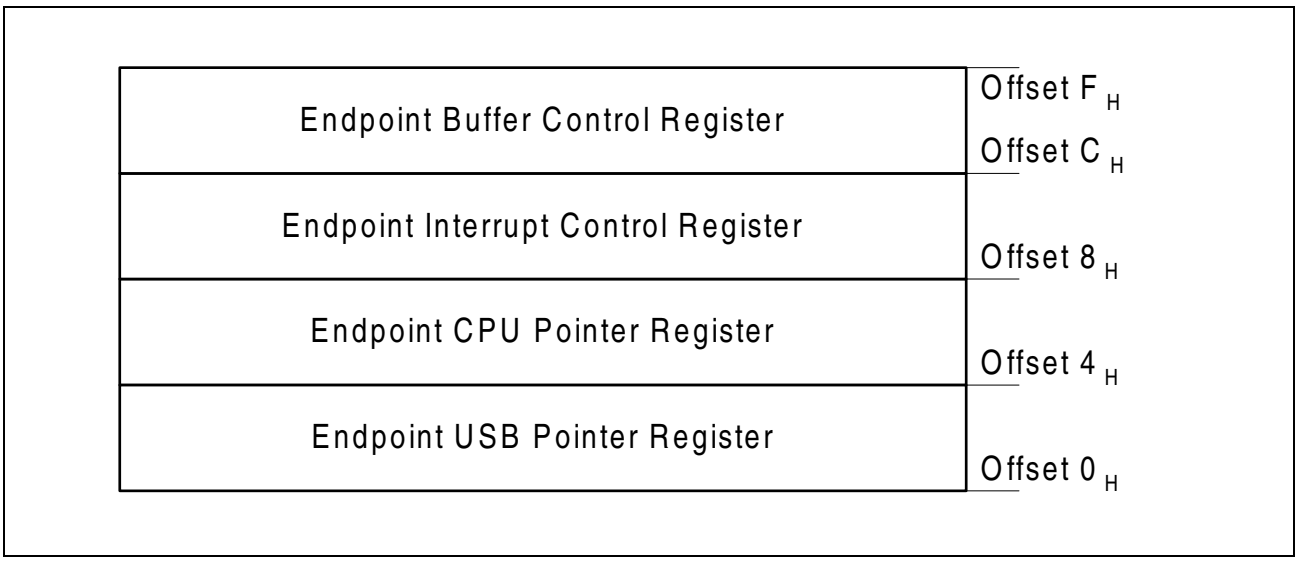


Figure 26-11 Organization of Endpoint Properties

26.1.2.10 USB Random Memory Access

In principal, the CPU may access each Memory location of the USB at any time.

However changing the Endpoint properties during runtime must be done carefully. The exact constraints are described in the paragraphs below.

If the automatic FIFO support is used and data within the Endpoint buffer area is changed during runtime by random memory accesses, the automatic FIFO support can be guarantied only when the application takes care to update the referring Endpoint properties that are affected by these accesses.

An update of the CPU pointer is necessary when the buffer has been filled up by the CPU beyond the CPU pointer, because this is the only way to inform the USB about the exact filling state of that buffer.

On the other hand, no pointer update is necessary, if only some data bytes in the buffer are changed without adding new data beyond the CPU pointer. This is helpful when a big packet is to be retransmitted several times with only some bits changed in each retransmission.

26.1.2.11 Reset Behavior

After any reset, the content of the Endpoint properties area of the internal RAM of the USB is undefined and must be set to the appropriate values by the application software. After power-up, the content of the Endpoint buffer area is reset to test values, while the Endpoint buffer area remains unchanged during a reset if the power supply is not turned off.

A reset operation of the UDC can only be achieved under software control. A hardware reset operation puts only the internal CPU interface of the USB module and its MMU into a well defined reset state.

The software reset, which must be executed after a hardware reset, is initiated by setting bit DCR.SWR by software. Bit DCR.SWR is reset automatically by hardware when the software reset operation of the USB module is finished. Furthermore, with the reset of bit DCR.SWR, bit DSR.DINIT is set indicating that CPU has to initialize the Endpoints of USB module. For the software reset, bus clock and USB clock must be turned on.

Bit DSR.DINIT is set automatically after the software reset has been performed.

26.1.2.12 Clock Generation

The USB must be provided by a single 48 MHz clock. This clock is handled separately from the Bus-Peripheral-Interface-Clock (BPI-Clock). A divide by 8 divider inside the module provides the 6 MHz clock for the low speed operation.

UDC Clock

The UDC operates at 12 MHz in full speed mode and 1.5 MHz in low speed mode. This clock is synchronized to the bit stream on the USB and it is generated within the UDC itself.

Proposal for the Separation of the two Clock Domains

The Assembly Buffer that interfaces to the UDC has two clock domains, i.e. the UDC 12/1.5 MHz and the BPI-Clock domain.

Full Speed Device

A full speed device handles data flow with a fixed data rate of 12 Mbit/s. Therefore, a user-provided clock signal of 48 MHz is required.

Low Speed Device

The USB transceiver supports Full Speed Device mode only.

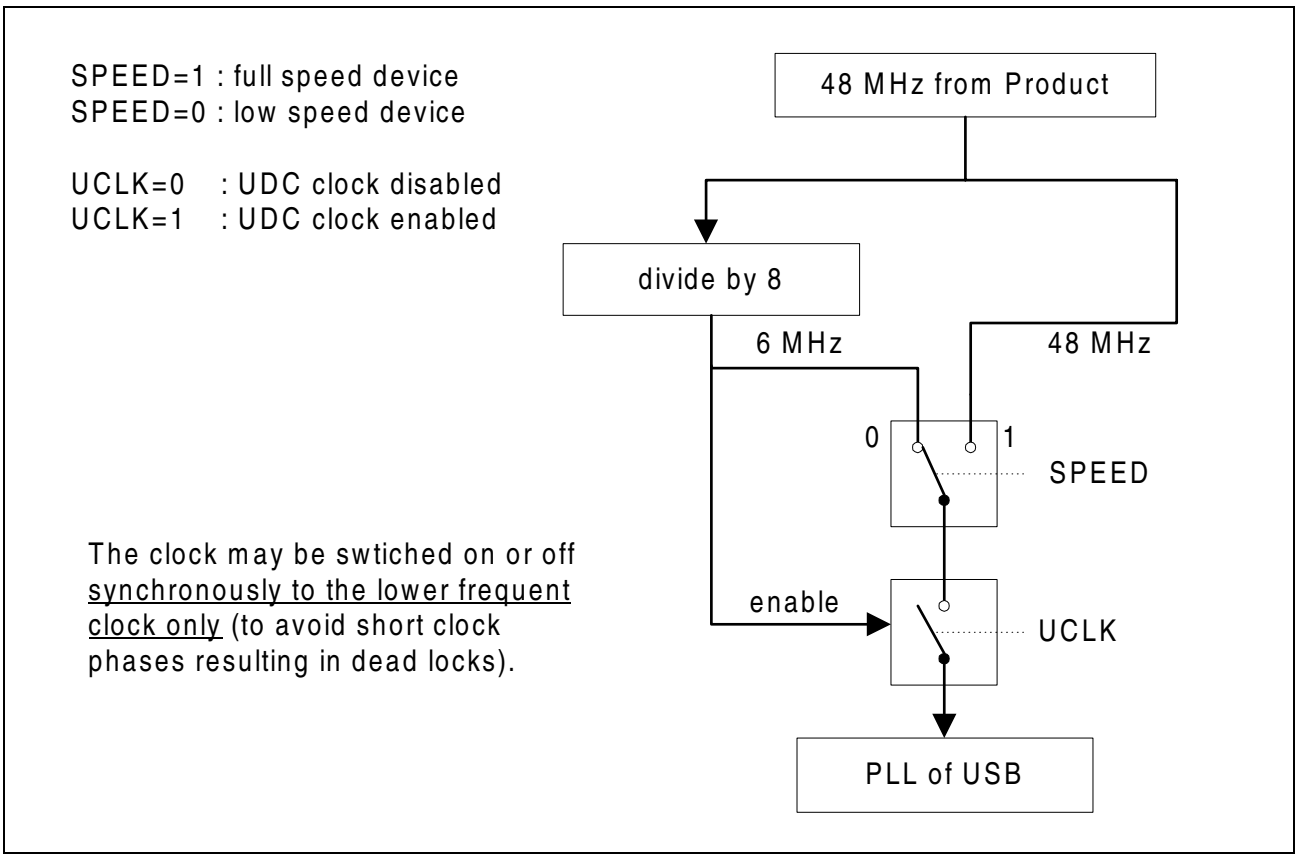

SPEED=1 : full speed device
SPEED=0 : low speed device
UCLK=0 : UDC clock disabled
UCLK=1 : UDC clock enabled
48 MHz from Product
divide by 8
6 MHz
48 MHz
0
1
SPEED
The clock may be swtiched on or off synchronously to the lower frequent clock only (to avoid short clock phases resulting in dead locks).
enable
UCLK
PLL of USB

Figure 26-12 Clock Control Block

26.1.2.13 Clock Control and Power Saving

The 48 MHz clock should be switched off (UCLK) for the power saving mode purpose if the module is not used by the application. In addition, the transceivers must be shut off separately (TPWD, RPWD). Switching off UCLK results in a reset of all module FFs/state machines.

Clock Rate Range

The clock rate range is 48 MHz and above. The minimum frequency may be lower depending on implementation details.

Suspend Mode Behavior

After the suspend request, the module performs a reset of all module FFs/internal state machines before acknowledging immediately the suspend mode. After the suspend mode, the module must be initialized completely.

26.2 USB Kernel Registers

Figure 26-13 and Table 26-5 show all registers associated with the USB Kernel.

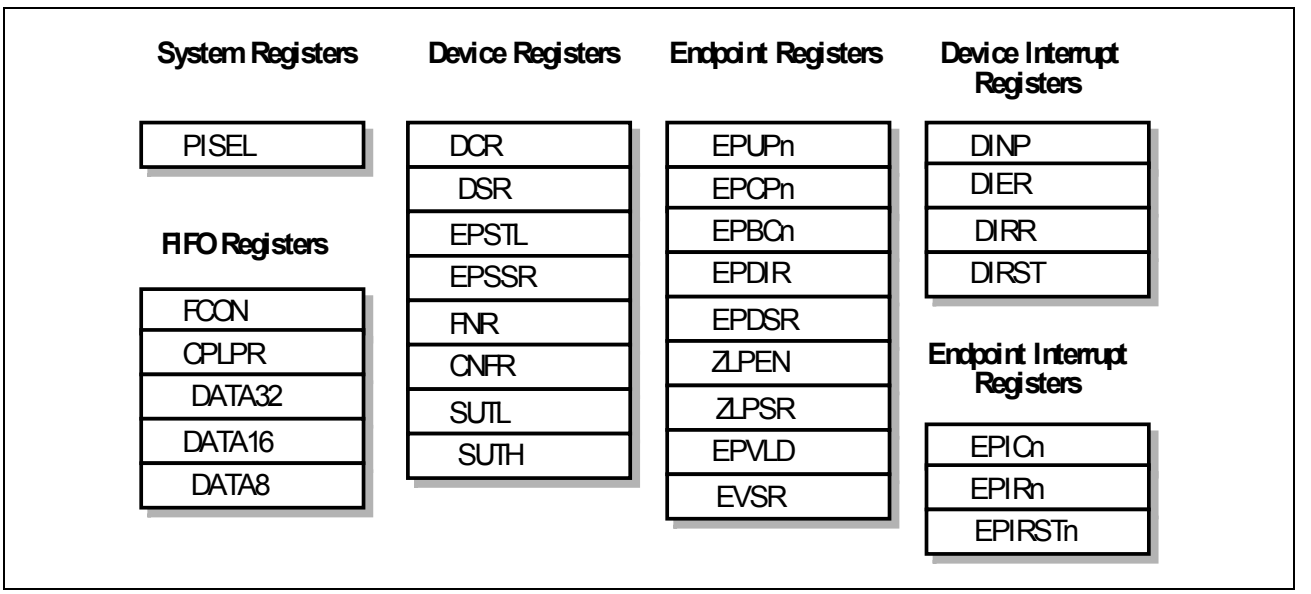


Figure 26-13 USB Kernel Registers

The USB has its own memory block. The two blocks – registers and memory – are concatenated to one module address space by using address offsets for each block:

- SFR_BASE
- RAM_BASE

Table 26-5 USB Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
SFR_BASE Registers of USB			
PISEL	Input Port Selection Register	SFR_BASE + 0004_H	Page 26-26
DCR	Device Control Register	SFR_BASE + 0010_H	Page 26-27
DSR	Device Status Register	SFR_BASE + 0014_H	Page 26-29
EPSTL	Endpoint Stall Register	SFR_BASE + 0018_H	Page 26-32
EPSSR	Endpoint Stall Set/Reset Register	SFR_BASE + $001C_H$	Page 26-33
CNFR	Configuration Request Register	SFR_BASE + 0020_H	Page 26-34
FNR	Frame Number Register	SFR_BASE + 0024_H	Page 26-34
EPDIR	Endpoint Direction Register	SFR_BASE + 0028_H	Page 26-40
EPDSR	Endpoint Direction Set/Reset Register	SFR_BASE + $002C_H$	Page 26-41

Table 26-5 USB Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
FCON	FIFO Control Register	SFR_BASE + 0030_H	Page 26-45
CPLPR	CPU Local Pointer Register	SFR_BASE + 0034_H	Page 26-48
DATA32	Data Register for 32-bit Accesses	SFR_BASE + 0038_H	Page 26-49
DATA16	Data Register for 16-bit Accesses	SFR_BASE + $003C_H$	Page 26-50
DATA8	Data Register for 8-bit Accesses	SFR_BASE + 0040_H	Page 26-50
EPVLD	Endpoint Valid Register	SFR_BASE + 0044_H	Page 26-43
EVSR	Endpoint Valid Set/Reset Register	SFR_BASE + 0048_H	Page 26-44
ZLPEN	Zero-Length-Package Enable Register	SFR_BASE + 0090_H	Page 26-42
ZLPSR	Zero-Length-Package Set/Reset Register	SFR_BASE + 0094_H	Page 26-42
DIER	Device Interrupt Enable Register	SFR_BASE + $004C_H$	Page 26-54
DIRR	Device Interrupt Request Register	SFR_BASE + 0050_H	Page 26-56
DIRST	Device Interrupt Reset Register	SFR_BASE + 0054_H	Page 26-58
DINP	Device Interrupt Node Pointer Register	SFR_BASE + 0058_H	Page 26-53
EPIR0	Endpoint Interrupt Request Register	SFR_BASE + $005C_H$	Page 26-63
EPIR1	Endpoint Interrupt Request Register	SFR_BASE + 0060_H	Page 26-63
EPIR2	Endpoint Interrupt Request Register	SFR_BASE + 0064_H	Page 26-63
EPIR3	Endpoint Interrupt Request Register	SFR_BASE + 0068_H	Page 26-63
EPIRST0	Endpoint Interrupt Reset Register	SFR_BASE + $006C_H$	Page 26-64
EPIRST1	Endpoint Interrupt Reset Register	SFR_BASE + 0070_H	Page 26-64
EPIRST2	Endpoint Interrupt Reset Register	SFR_BASE + 0074_H	Page 26-64
EPIRST3	Endpoint Interrupt Reset Register	SFR_BASE + 0078_H	Page 26-64

Table 26-5 USB Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
RAM based Registers of Universal Serial Bus (USB)			
SUTL	Setup Token Low Bytes Register	RAM_BASE + 0000_H	**Page 26-35**
SUTH	Setup Token High Bytes Register	RAM_BASE + 0004_H	**Page 26-35**
EPUP0	Endpoint 0 USB Pointer Register	RAM_BASE + 0010_H	**Page 26-37**
EPCP0	Endpoint 0 CPU Pointer Register	RAM_BASE + 0014_H	**Page 26-38**
EPIC0	Endpoint 0 Interrupt Control Register	RAM_BASE + 0018_H	**Page 26-60**
EPBC0	Endpoint 0 Buffer Control Register	RAM_BASE + $001C_H$	**Page 26-39**
EPUP1	Endpoint 1 USB Pointer Register	RAM_BASE + 0020_H	**Page 26-37**
EPCP1	Endpoint 1 CPU Pointer Register	RAM_BASE + 0024_H	**Page 26-38**
EPIC1	Endpoint 1 Interrupt Control Register	RAM_BASE + 0028_H	**Page 26-60**
EPBC1	Endpoint 1 Buffer Control Register	RAM_BASE + $002C_H$	**Page 26-39**
EPUP2	Endpoint 2 USB Pointer Register	RAM_BASE + 0030_H	**Page 26-37**
EPCP2	Endpoint 2 CPU Pointer Register	RAM_BASE + 0034_H	**Page 26-38**
EPIC2	Endpoint 2 Interrupt Control Register	RAM_BASE + 0038_H	**Page 26-60**
EPBC2	Endpoint 2 Buffer Control Register	RAM_BASE + $003C_H$	**Page 26-39**
EPUP3	Endpoint 3 USB Pointer Register	RAM_BASE + 0040_H	**Page 26-37**
EPCP3	Endpoint 3 CPU Pointer Register	RAM_BASE + 0044_H	**Page 26-38**
EPIC3	Endpoint 3 Interrupt Control Register	RAM_BASE + 0048_H	**Page 26-60**
EPBC3	Endpoint 3 Buffer Control Register	RAM_BASE + $004C_H$	**Page 26-39**
EPUP4	Endpoint 4 USB Pointer Register	RAM_BASE + 0050_H	**Page 26-37**
EPCP4	Endpoint 4 CPU Pointer Register	RAM_BASE + 0054_H	**Page 26-38**
EPIC4	Endpoint 4 Interrupt Control Register	RAM_BASE + 0058_H	**Page 26-60**

Table 26-5 USB Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
EPBC4	Endpoint 4 Buffer Control Register	RAM_BASE + $005C_H$	Page 26-39
EPUP5	Endpoint 5 USB Pointer Register	RAM_BASE + 0060_H	Page 26-37
EPCP5	Endpoint 5 CPU Pointer Register	RAM_BASE + 0064_H	Page 26-38
EPIC5	Endpoint 5 Interrupt Control Register	RAM_BASE + 0068_H	Page 26-60
EPBC5	Endpoint 5 Buffer Control Register	RAM_BASE + $006C_H$	Page 26-39
EPUP6	Endpoint 6 USB Pointer Register	RAM_BASE + 0070_H	Page 26-37
EPCP6	Endpoint 6 CPU Pointer Register	RAM_BASE + 0074_H	Page 26-38
EPIC6	Endpoint 6 Interrupt Control Register	RAM_BASE + 0078_H	Page 26-60
EPBC6	Endpoint 6 Buffer Control Register	RAM_BASE + $007C_H$	Page 26-39
EPUP7	Endpoint 7 USB Pointer Register	RAM_BASE + 0080_H	Page 26-37
EPCP7	Endpoint 7 CPU Pointer Register	RAM_BASE + 0084_H	Page 26-38
EPIC7	Endpoint 7 Interrupt Control Register	RAM_BASE + 0088_H	Page 26-60
EPBC7	Endpoint 7 Buffer Control Register	RAM_BASE + $008C_H$	Page 26-39
EPUP8	Endpoint 8 USB Pointer Register	RAM_BASE + 0090_H	Page 26-37
EPCP8	Endpoint 8 CPU Pointer Register	RAM_BASE + 0094_H	Page 26-38
EPIC8	Endpoint 8 Interrupt Control Register	RAM_BASE + 0098_H	Page 26-60
EPBC8	Endpoint 8 Buffer Control Register	RAM_BASE + $009C_H$	Page 26-39
EPUP9	Endpoint 9 USB Pointer Register	RAM_BASE + $00A0_H$	Page 26-37
EPCP9	Endpoint 9 CPU Pointer Register	RAM_BASE + $00A4_H$	Page 26-38
EPIC9	Endpoint 9 Interrupt Control Register	RAM_BASE + $00A8_H$	Page 26-60
EPBC9	Endpoint 9 Buffer Control Register	RAM_BASE + $00AC_H$	Page 26-39

Table 26-5 USB Kernel Registers (cont'd)

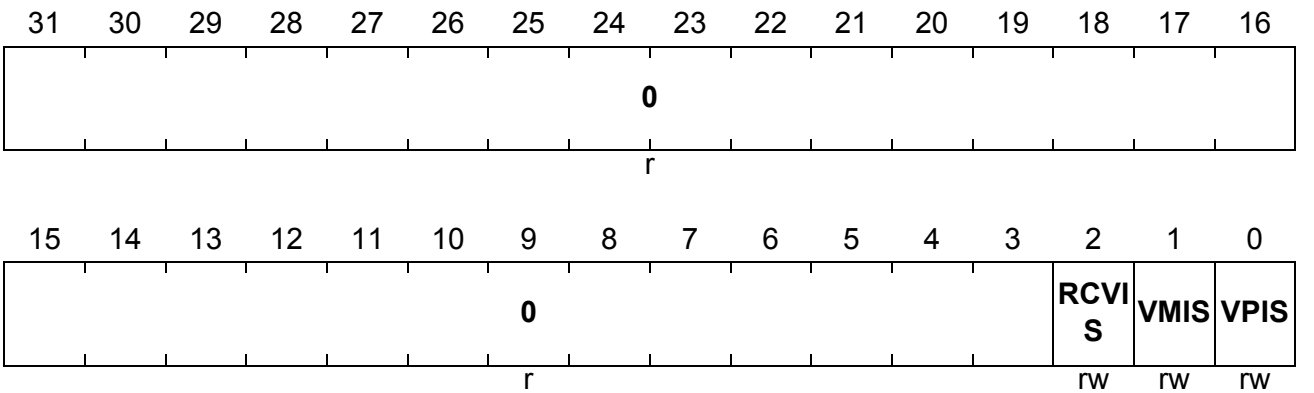
Register Short Name	Register Long Name	Offset Address	Description see
EPUP10	Endpoint 10 USB Pointer Register	RAM_BASE + $00B0_H$	Page 26-37
EPCP10	Endpoint 10 CPU Pointer Register	RAM_BASE + $00B4_H$	Page 26-38
EPIC10	Endpoint 10 Interrupt Control Register	RAM_BASE + $00B8_H$	Page 26-60
EPBC10	Endpoint 10 Buffer Control Register	RAM_BASE + $00BC_H$	Page 26-39

26.2.1 System Registers

For switching between different port input/output sources the USB unit provides an input multiplexer. This multiplexer allows the selection between two input sources.

The PISEL register is used to select between internal and external USB transceiver.

PISEL
Input Select Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
VPIS	0	rw	**Select Internal or External Transceiver for VPI** 0 Internal transceiver 1 External transceiver
VMIS	1	rw	**Select Internal or External Transceiver for VMI** 0 Internal transceiver 1 External transceiver
RCVIS	2	rw	**Select Internal or External Transceiver for RCVI** 0 Internal transceiver 1 External transceiver
0	[31:3]	r	**Reserved;** read as 0; should be written with 0.

26.2.2 Device Registers

DCR
Device Control Register DCR **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPU SRL	USB SRL	0		DADDI EN	DADDI DIS	SPEEDS	SWRS	EOID	RSM	UCLKR	UCLKS	TPWDR	TPWDS	RPWDR	RPWDS
w	w	r		w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RPWDS	0	w	**USB Receiver Power Down Set** 0 No action occurs - Receiver remains in Power Down Mode 1 Receiver is enabled *Note: If bits RPWDS and RPWDR are set at the same time, DSR.RPWD will not change.*
RPWDR	1	w	**USB Receiver Power Down Reset** 0 No action occurs 1 Receiver is disabled *Note: If bits RPWDS and RPWDR are set at the same time, DSR.RPWD will not change.*
TPWDS	2	w	**USB Transmitter Power Down Set** 0 No action occurs - Transceiver remains in Power Down Mode 1 Transceiver is enabled *Note: If bits TPWDS and TPWDR are set at the same time, DSR.TPWD will not change.*
TPWDR	3	w	**USB Transmitter Power Down Reset** 0 No action occurs 1 Transceiver is disabled *Note: If bits TPWDS and TPWDR are set at the same time, DSR.TPWD will not change.*

Field	Bits	Type	Description
UCLKS	4	w	**UDC Clock Enable** 0 No action is taken 1 Bit DSR.UCLK is set *Note: If bits UCLKS and UCLKR are set at the same time, DSR.UCLK will not change.*
UCLKR	5	w	**UDC Clock Disable** 0 No action is taken 1 Bit DSR.UCLK is reset *Note: If bits UCLKS and UCLKR are set at the same time, DSR.UCLK will not change.*
RSM	6	w	**Resume Bus Activity** 0 No action is taken 1 When the USB device is in suspend mode, setting bit RSM resumes bus activity of the device. In response to this action, the USB will disassert the suspend bit automatically and will perform the remote wake-up operation. After the suspend mode is left, the suspend end interrupt SEI is generated.
EOID	7	w	**End of Initialization Data** 0 No action is taken 1 The USB assumes that the last byte of the initialization data has been written to the RAM by the CPU and can now be read by the UDC.
SWRS	8	w	**Software Reset** 0 No action is taken 1 A software reset is generated Please refer to **Chapter 26.1.2.6**. *Note: This bit can only be written with DSR.UCLK = 1 (USB Clock enabled).*
SPEEDS	9	w	**Low/Full Speed Select Set** 0 No action is taken 1 Bit DSR.SPEED is set *Note: Bit protection for bit DSR.SPEED. This bit can only be written with bit SWRS = 1 (software reset).*

Field	Bits	Type	Description
DADDIDIS	10	w	**Device Attached Signal Disabled** If not set (default) an external pin/signal is used to detect device attached. If set the device attached signal is no longer a separate input pin but is generated by a detection circuit evaluating D+ / D- signals.
DADDIEN	11	w	**Device Attached Signal Enabled** 0 No action is taken 1 An external pin/signal is used to detect device attached (DSR.DADDIDIS is reset)
USBSRL	14	w	**USB Shadow Register Reload** 0 No action is taken 1 USB shadow registers are reloaded
CPUSRL	15	w	**CPU Shadow Register Reload** 0 No action is taken 1 CPU shadow registers are reloaded on the next Host transmission
0	[13:12], [31:16]	r	**Reserved;** read as 0; should be written with 0.

Note: CPUSRL and USBSRL are implemented to reload the EPCP register and the EPUP register (which are called “shadow register” because they are stored inside the memory). These two bits are used when a reload needs to be forced to change the endpoint properties.

DSR

Device Status Register DSR **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								TIP	0			UEP			
r								rh	r			rh			

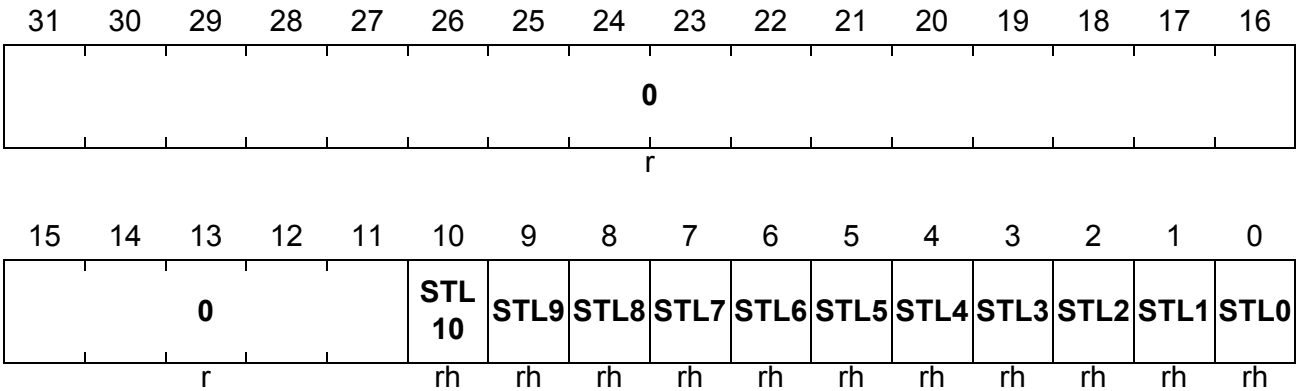
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		DADDIDIS	DIR	0		SPEED	SWR	DINIT	SUSP	0	UCLK	0	TPWD	0	RPWD
r		rh	rh	r		rh	rh	rh	rh	r	rh	r	rh	r	rh

Field	Bits	Type	Description
RPWD	0	rh	**USB Receiver Power Down** 0 The USB receiver is in Power Down Mode 1 The USB receiver is active
TPWD	2	rh	**USB Transmitter Power Down** 0 The USB transmitter is in Power Down Mode 1 The USB transmitter is active
UCLK	4	rh	**UDC Clock Enable** Bit UCLK controls the USB Device Core clock (48 MHz in full speed mode, 6 MHz in low speed mode). 0 UDC clock is disabled. 1 UDC clock is enabled. *Note: If the USB is not used, UCLK should be set to 0 (default) for power saving reasons.*
SUSP	6	rh	**Suspend Mode** This bit is set when the USB is idle for more than 3 ms. It will remain set until there is a non idle state on the USB cable or when bit DCR.RSM is set.
DINIT	7	rh	**Device Initialization in Progress** At the end of a software reset, bit DINIT is set by hardware. After software reset of the USB module, the USB module must be initialized by the CPU. When DINIT is set after a software reset, 6 bytes of each Endpoint must be written to the Memory starting with writing byte 0 to the address 0 and then incrementing the address for each byte. After the last byte (6 × n, n = number of Endpoints) has been written, bit DINIT is reset by software after a successful initialization sequence.
SWR	8	rh	**Software Reset** Setting bit SWR initiates a software reset operation of the USB device. This bit is cleared by hardware after successful reset operation. SWR cannot be reset by software.

Field	Bits	Type	Description
SPEED	9	rh	**Low/Full Speed Select** Bit SPEED configures the USB module for full speed (12 MBaud) or low speed (1.5 MBaud) mode. This bit can only be written with bit SWR = 1 (software reset). After hardware reset the USB module runs in low speed. 0 Low speed mode selected (default after reset) 1 Full speed mode selected *Note: As low speed is not supported in TC1130, this bit must be set as 1.*
DIR	12	rh	**Direction of Endpoint 0** Bit DIR reflects the direction of the last USB access to Endpoint 0, independent whether EPVLD.EPV0 has been set or not. 0 WRITE 1 READ
DADDIDIS	13	rh	**Device Attached Signal Disabled** 0 (default) An external pin/signal is used to detect device attached 1 The device attached signal is no longer a separate input pin but is generated by a detection circuit evaluating D+/D- signals.
UEP	[19:16]	rh	**USB Endpoint in Use** 0 No effect 1 The Endpoint is currently accessed by the USB *Note: The application software should not access the USB Endpoint in use by random access to the RAM. If the application software preforms any RAM access to the USB Endpoint in use, the application software must realize EPCP.CAPn and EPBC.NEBn.*
TIP	23	rh	**Transfer in Progress** 0 No effect 1 USB host currently accesses the device
0	1, 3, 5, [11:10] [15:14] [22:20] [31:24]	r	**Reserved;** read as 0; should be written with 0.

The EPSTL register is used to signal to the USB, if an Endpoint is active.

EPSTL
Endpoint Stall Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
STLn (n = 0 … 10)	[10:0]	rh	**Endpoint n Stall** 0 Endpoint n is active 1 Endpoint n is stalled *Note: If the Endpoint stall bit for Endpoint 0 (STL0) is set, the next incoming setup token will automatically clear it.*
0	[31:11]	r	**Reserved;** read as 0; should be written with 0.

The EPSSR register is used to set/reset the EPSTL.STLn bits.

EPSSR
Endpoint Stall Set/Reset Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					STLR10	STLR9	STLR8	STLR7	STLR6	STLR5	STLR4	STLR3	STLR2	STLR1	STLR0
r					w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					STLS10	STLS9	STLS8	STLS7	STLS6	STLS5	STLS4	STLS3	STLS2	STLS1	STLS0
r					w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
STLSn (n = 0 … 10)	[10:0]	w	**Set Endpoint n Stall** 0 No effect 1 EPSTL.STLn is set to 1 *Note: If bits STLSn and STLRn are set at the same time, EPSTL.STLn will not change.*
STLRn (n = 0 … 10)	[26:16]	w	**Reset Endpoint n Stall** 0 No effect 1 EPSTL.STLn is set to 0 *Note: If bits STLSn and STLRn are set at the same time, EPSTL.STLn will not change.*
0	[15:11]; [31:27]	r	**Reserved;** read as 0; should be written with 0.

FNR
Frame Number Register FNR

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					FNR 10	FNR 9	FNR 8	FNR 7	FNR 6	FNR 5	FNR 4	FNR 3	FNR 2	FNR 1	FNR 0
r					rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
FNRn (n = 0 … 10)	[10:0]	rh	**Frame Number Value** FNR[10:0] hold the current 11-bit frame number of the latest SOF token.
0	[31:11]	r	**Reserved;** read as 0; should be written with 0.

CNFR
Configuration Request Register CNFR

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
t															

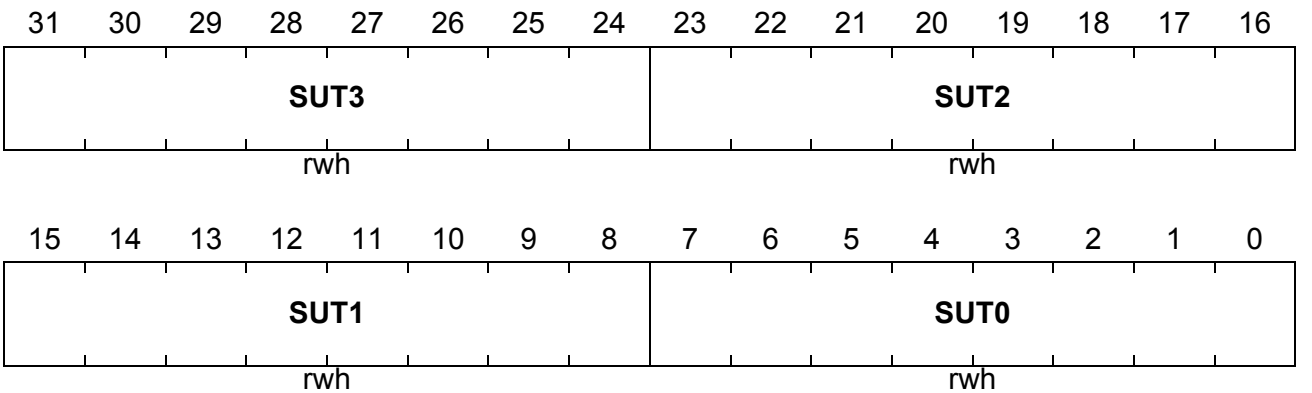
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF3	CF2	CF1	CF0	0				IF3	IF2	IF1	IF0	AS3	AS2	AS1	AS0
rh	rh	rh	rh	r				rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
AS[3:0]	[3:0]	rh	**Alternate Setting** Selects one of 16 Alternate Settings (AS 0 - 15).
IF[3:0]	[7:4]	rh	**Interface Number** Selects one of 16 Interfaces (IF 0 - 15).
CF[3:0]	[15:12]	rh	**Configuration Value** Selects one of 16 Configurations (CF 0 - 15).

Field	Bits	Type	Description
0	[11:8], [31:16]	r	**Reserved;** read as 0; should be written with 0.

SUTL

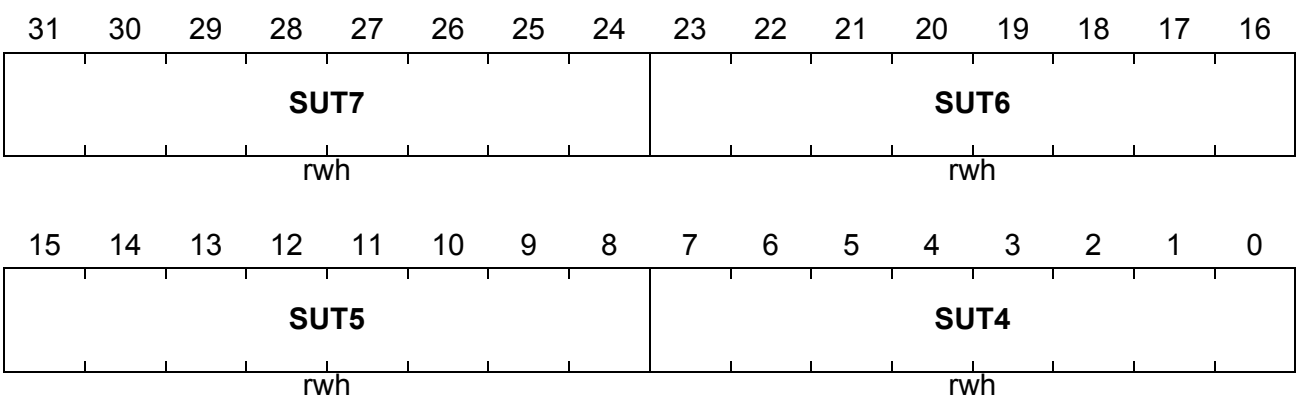
Setup Token Low Bytes Register SUTL **Reset Value: XXXX XXXX$_H$**



Field	Bits	Type	Description
SUTn (n = 0 … 3)	[31:0]	rwh	**Setup Token Byte n** These 4 bytes hold the value that is sent from the USB host in the “Setup Token”. The function of these bits is completely determined by the USB specification.

SUTH

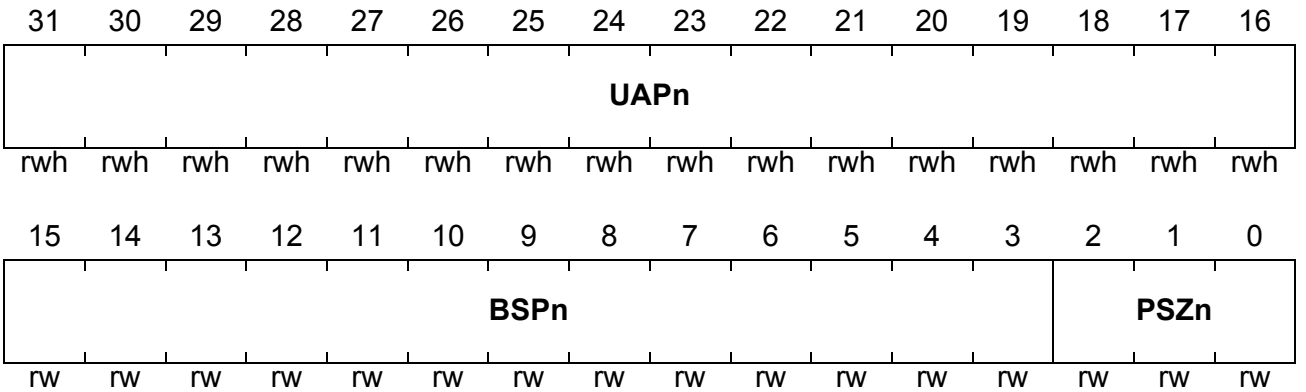
Setup Token High Bytes Register SUTH **Reset Value: XXXX XXXX$_H$**



Field	Bits	Type	Description
SUTn (n = 4 … 7)	[31:0]	rwh	**Setup Token Byte n** These 4 bytes hold the value that is sent from the USB host in the “Setup Token”. The function of these bits is completely determined by the USB specification.

26.2.3 Endpoint Registers

EPUPn (n = 0 … 10)
Endpoint n USB Pointer Register **Reset Value: XXXX XXXX$_H$**


31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
UAPn
rwh rwh rwh rwh rwh rwh rwh rwh rwh rwh rwh rwh rwh rwh rwh rwh
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
BSPn
PSZn
rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw

Field	Bits	Type	Description
PSZn (n = 0 … 10)	[2:0]	rw	**Packet Size for Endpoint n** The size of the packets of Endpoint n. 000 Packet size is 8 bytes 001 Packet size is 16 bytes 010 Packet size is 32 bytes 011 Packet size is 64 bytes 100 Packet size is 128 bytes 101 Packet size is 256 bytes 110 Packet size is 512 bytes 111 Packet size is 1024 bytes
BSPn (n = 0 … 10)	[15:3]	rw	**Base Pointer for Endpoint n** The base of the buffer for Endpoint n. The granularity is 8 bytes. The 3 LSBs are regarded as 0.
UAPn (n = 0 … 10)	[31:16]	rwh	**USB Access Pointer for Endpoint n** The current value of the USB pointer for Endpoint n.

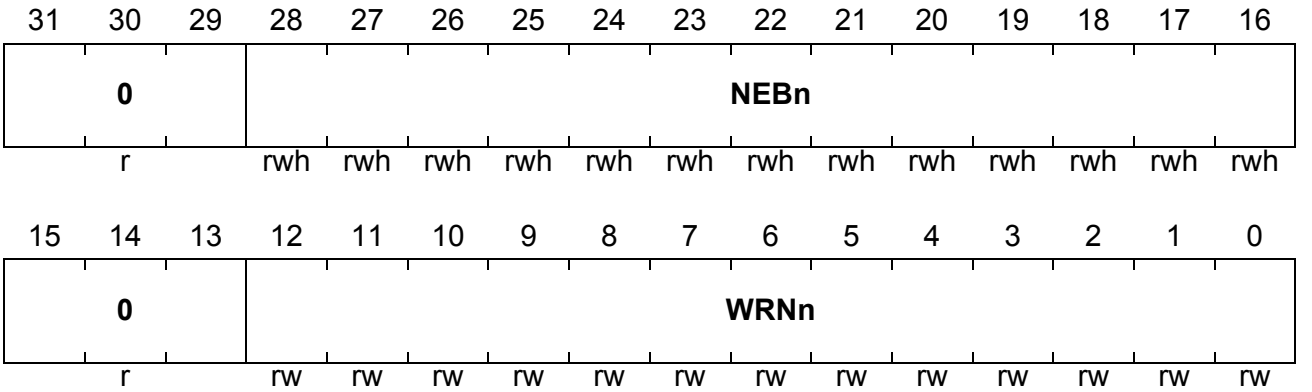
EPCPn (n = 0 … 10)
Endpoint n CPU Pointer Register **Reset Value: XXXX XXXX$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CAPn															
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BSZn													0		UAV Mn
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r		rwh

Field	Bits	Type	Description
UAVMn (n = 0 … 10)	0	rwh	**Automatic USB End Point Valid Update** Bit UAVMn enables the automatic update of the EPVLD.EPVn bits (see register EPVLD). • USB Read Access: If the CPU has at least 1 packet within current transfer, EPVLD.EPVn is set automatically. • USB Write Access: If at least one packet buffer is empty in current transfer, EPVLD.EPVn is set automatically. EPVLD.EPVn is reset, if no empty packet buffer is available. *Note: UAVMn is not reset on EOT.*
BSZn (n = 0 … 10)	[15:3]	rw	**Buffer Size for Endpoint n** The buffer size for Endpoint n is calculated as (EPUPn.PSZ$_n$) × BSZ$_n$
CAPn (n = 0 … 10)	[31:16]	rwh	**CPU Access Pointer for Endpoint n** The current value of the CPU pointer for Endpoint n.
0	[2:1]	r	**Reserved;** read as 0; should be written with 0.

EPBCn (n = 0 … 10)
Endpoint n Buffer Control Register **Reset Value: XXXX XXXX$_H$**


31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
0
NEBn
r
rwh rwh rwh rwh rwh rwh rwh rwh rwh rwh rwh rwh rwh
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0
WRNn
r
rw rw rw rw rw rw rw rw rw rw rw rw rw

Field	Bits	Type	Description
WRNn (n = 0 … 10)	[12:0]	rw	**Warning Level for Endpoint n** The value in WRNn is compared to the value in NEBn. A warning level interrupt is generated on equality if the interrupt is enabled.
NEBn (n = 0 … 10)	[28:16]	rwh	**Number of Empty Packet Buffers for Endpoint n** The number of empty Packet Buffers for Endpoint n. It is essential for the correct function of USB that NEBn is initialized by software, in general with the value of EPCPn.BSZn. *Note: If the warning level is reached, a warning level interrupt is generated. After serving that interrupt, the application software may check NEBn for controlling whether the filling status of the Endpoint buffer still exceeds the warning level or not. This prevents the CPU from a dead lock situation if the data transfer speed is not predictable.*
0	[15:13], [31:29]	r	**Reserved;** read as 0; should be written with 0.

The EPDIR register is used to signal to the USB, if the Endpoint is configured for an USB write access or for an USB read access.

For Endpoint 0, any packet direction can be accepted, because this Endpoint is always bidirectional. If the present direction selection in EPDIR.DIR0 does not match the direction of an incoming packet for Endpoint 0, EPDIR.DIR0 must be changed by the application software. In addition, EPVLD.EPV0 must be set before data is transferred for any other Endpoints.

EPDIR
Endpoint Direction Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					DIR 10	DIR9	DIR8	DIR7	DIR6	DIR5	DIR4	DIR3	DIR2	DIR1	DIR0
r					rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
DIRn (n = 0 … 10)	[10:0]	rh	**Endpoint n Direction** 0 The data flow for endpoint n is from host to CPU 1 The data flow for endpoint n is from CPU to host *Note: It is not allowed to change the direction during a packet transfer.* *Note: Endpoint 0 is the bidirectional control Endpoint.*
0	[31:11]	r	**Reserved;** read as 0; should be written with 0.

The EPDSR register is used to set/reset the EPDIR.DIRn bits.

EPDSR
Endpoint Direction Set/Reset Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					DIR R10	DIR R9	DIR R8	DIR R7	DIR R6	DIR R5	DIR R4	DIR R3	DIR R2	DIR R1	DIR R0
r					w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					DIR S10	DIR S9	DIR S8	DIR S7	DIR S6	DIR S5	DIR S4	DIR S3	DIR S2	DIR S1	DIR S0
r					w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
DIRSn (n = 0 … 10)	[10:0]	w	**Set Endpoint n Direction** 0 No effect 1 EPDIR.DIRn is set to one *Note: If bits DIRSn and DIRRn are set at the same time, EPDIR.DIRn will not change.*
DIRRn (n = 0 … 10)	[26:16]	w	**Reset Endpoint n Direction** 0 No effect 1 EPDIR.DIRn is set to 0 *Note: If bits DIRSn and DIRRn are set at the same time, EPDIR.DIRn will not change.*
0	[15:11]; [31:27]	r	**Reserved;** read as 0; should be written with 0.

The ZLPEN register is used to signal to the USB whether a zero-length-package has to be sent. If ZLPEN.ZLPn is 1 and EPVLD.EPVn is 0, a zero-length-package is sent with the next transfer.

ZLPEN
Zero-Length-Package Enable Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					ZLP 10	ZLP 9	ZLP 8	ZLP 7	ZLP 6	ZLP 5	ZLP 4	ZLP 3	ZLP 2	ZLP 1	ZLP 0
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
ZLPn (n = 0 … 10)	[10:0]	rh	**Zero-Length-Package n** If ZLPEN.ZLPn is 1 and EPVLD.EPVn is 0, a zero-length-package is sent with the next transfer. Thereafter the ZLPEN.ZLPn is reset.
0	[31:11]	r	**Reserved;** read as 0; should be written with 0.

The ZLPSR register is used to set/reset the ZLPEN.ZLPn bits.

ZLPSR
Zero-Length-Package Set/Reset Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					ZLP R10	ZLP R9	ZLP R8	ZLP R7	ZLP R6	ZLP R5	ZLP R4	ZLP R3	ZLP R2	ZLP R1	ZLP R0
r					w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					ZLP S10	ZLP S9	ZLP S8	ZLP S7	ZLP S6	ZLP S5	ZLP S4	ZLP S3	ZLP S2	ZLP S1	ZLP S0
r					w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
ZLPSn (n = 0 … 10)	[10:0]	w	**Zero-Length-Package Set n** 0 No effect 1 ZLPEN.ZLPn is set to 1
ZLPRn (n = 0 … 10)	[26:16]	w	**Zero-Length-Package Reset n** 0 No effect 1 ZLPEN.ZLPn is set to 0
0	[15:11]; [31:27]	r	**Reserved;** read as 0; should be written with 0.

The EPVLD register is used to signal to the USB, if an Endpoint is ready for transfer and whether a zero-length-package has to be sent. If USB requires data transfer for a certain Endpoint and the referring bit EPVLD.EPVn is set to zero, the USB automatically sends NACK. If the referring bit EPVLD.EPVn is 0 and ZLPEN.ZLPn is 1, a zero-length-package is sent. If the referring bit EPVLD.EPVn is set to 1 but ZLPEN.ZLPn is 0, data will be transferred automatically to and from the buffer of that Endpoint. If UAVMn is set, EPVn is set automatically reflecting if the filling status of the buffer does allow for the next USB access.

EPV = 0 and ZLP = 0 → NACK is sent
EPV = 0 and ZLP = 1 → ZLP is sent
EPV = 1 and ZLP = 0 → data transfer
EPV = 1 and ZLP = 1 → data transfer; if EPV reset, a ZLP is sent

EPVLD.EPV0 will be reset automatically together with the setting of DIRR.SUI. EPVLD.EPVn will be reset automatically together with the detection of EOT for Endpoint n.

EPVLD
Endpoint Valid Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					EPV 10	EPV 9	EPV 8	EPV 7	EPV 6	EPV 5	EPV 4	EPV 3	EPV 2	EPV 1	EPV 0
r					rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
EPVn (n = 0 … 10)	[10:0]	rh	**USB Endpoint n Valid** 0 Endpoint n is not ready for transfer (NACK/ZLP) 1 Endpoint n is ready for transfer (ACK).
0	[31:11]	r	**Reserved;** read as 0; should be written with 0.

The EVSR register is used to set/reset the EPVLD.EPVn bits.

EVSR
Endpoint Valid Set/Reset Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					EPVR10	EPVR9	EPVR8	EPVR7	EPVR6	EPVR5	EPVR4	EPVR3	EPVR2	EPVR1	EPVR0
r					w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					EPVS10	EPVS9	EPVS8	EPVS7	EPVS6	EPVS5	EPVS4	EPVS3	EPVS2	EPVS1	EPVS0
r					w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
EPVSn (n = 0 … 10)	[10:0]	w	**Set USB Endpoint n Valid** 0 No effect 1 EPVLD.EPVn is set to 1 *Note: If bits EPVSn and EPVRn are set at the same time, EPVLD.EPVn will not change.*
EPVRn (n = 0 … 10)	[26:16]	w	**Reset USB Endpoint n Valid** 0 No effect 1 EPVLD.EPVn is set to 0 *Note: If bits EPVSn and EPVRn are set at the same time, EPVLD.EPVn will not change.*
0	[15:11]; [31:27]	r	**Reserved;** read as 0; should be written with 0.

26.2.4 FIFO Registers

The FCON register contains the control bits for the FIFO.

FCON
FIFO Control Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					EPFVLD10	EPFVLD9	EPFVLD8	EPFVLD7	EPFVLD6	EPFVLD5	EPFVLD4	EPFVLD3	EPFVLD2	EPFVLD1	EPFVLD0
r					rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					CLREP	DONE	FVLD	0				CEPS			
r					w	w	rh	r				rw			

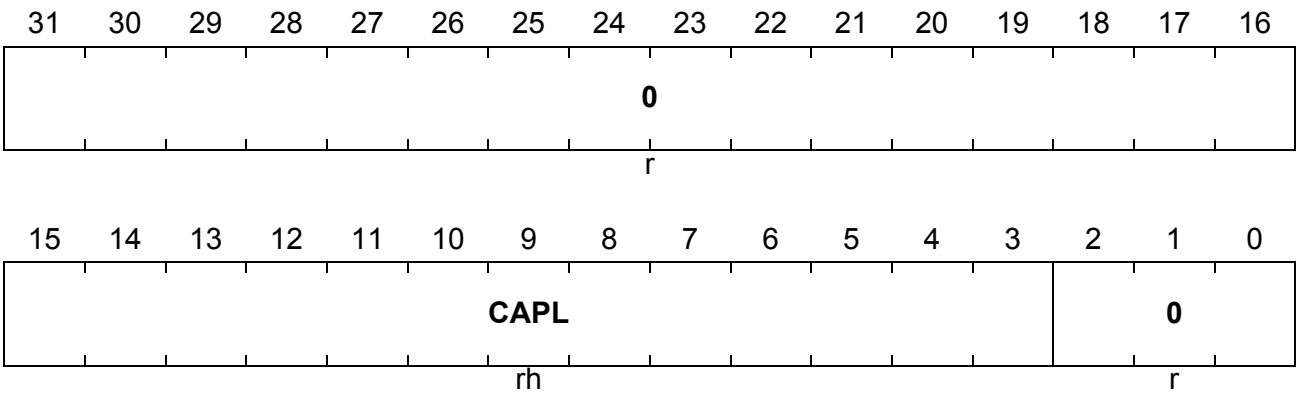
Field	Bits	Type	Description
CEPS[3:0]	[3:0]	rw	**CPU Endpoint Select** 0000 Endpoint 0 is selected 0001 Endpoint 1 is selected 0010 Endpoint 2 is selected 0011 Endpoint 3 is selected 0100 Endpoint 4 is selected 0101 Endpoint 5 is selected 0110 Endpoint 6 is selected 0111 Endpoint 7 is selected 1000 Endpoint 8 is selected 1001 Endpoint 9 is selected 1010 Endpoint 10 is selected 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved For normal use, the Endpoint is changed by the application only if a packet is completely transferred to or from the Endpoint buffer or if bit DONE has been set.

Field	Bits	Type	Description
FVLD	8	rh	**FIFO Valid** 0 FIFO is not ready for transfer 1 FIFO is ready for transfer Data that is written to a DATA32/16/8 register before FVLD is set will be discarded. This is signaled to the application by generating the DIRR.FNRI (FIFO is not ready for interrupt).
DONE	9	w	**Buffer done by CPU** • USB Read Access: If bit DONE is set, the USB assumes that the last byte for that transfer has been written. This allows the UDC to access this packet although it is not completely written. An End of Transfer Condition will be signaled automatically to the USB. The CPU must not fill further buffers during the pending transfer, as EPCP.CAPn points to the last byte to be transferred. • USB Write Access: If bit DONE is set, the USB assumes that the last byte for that transfer has been read. Thus the USB moves the CPLPR.CAPL to the base of the next packet, writes it back to EPCP.CAPn and assumes that this packet is empty. (“Skip” functionality.)

Field	Bits	Type	Description
CLREP	10	w	**Clear Endpoint** The Endpoint clear bit is used to reset the buffer and the control bits of the Endpoint FIFO on both sides: the CPU and the UDC part. (Thus the current transfer between Host and UDC are cancelled immediately.) Only the currently loaded Endpoints can be cleared. I.e. if CLREP is set, CEPS is disregarded in that write cycle. The bits FCON.FVLD and EPVLD.EPVn will be reset when CLREP is set. Setting of bit CLREP does not change the direction of Endpoint n. This means, bit DSR.DIRn is not changed. *Note: After setting bit CLREP the CPU access pointer CPLPR.CAPL and USB access pointer EPUP.UAPn must be set to the base pointer EPUP.BSPn by the application software. EPBC.NEBn may need to be updated (with the value of EPCR.BSZn) by application software. (n = CEPS in this bit description).*
EPFVLDn (n = 0 … 10)	[26:16]	rh	**Endpoint FIFO Valid** The Endpoint FIFO valid bit always indicates if the data buffer for the Endpoint n is ready for data transfer between USB and CPU. It can be used by the software as a simple signal, whether the Endpoint should be selected by CEPS or not. Write Data is available for the CPU Read Memory space is available for the CPU
0	[7:4], [15:11], [31:27]	r	**Reserved;** read as 0; should be written with 0.

The CPLPR register contains the current CPU Access Pointer that is located in the FIFO. The 3 LSBs are not realized because the internal memory is organized in 32 bit which is accessed via the assembly buffer. This results in a uncertainty of 8 bytes for CAPL.

CPLPR
CPU Local Pointer Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CAPL	[15:3]	rh	**CPU Local Access Pointer** The current CPU Access Pointer that is locally held in the FIFO.
0	[2:0] [31:16]	r	**Reserved;** read as 0; should be written with 0.

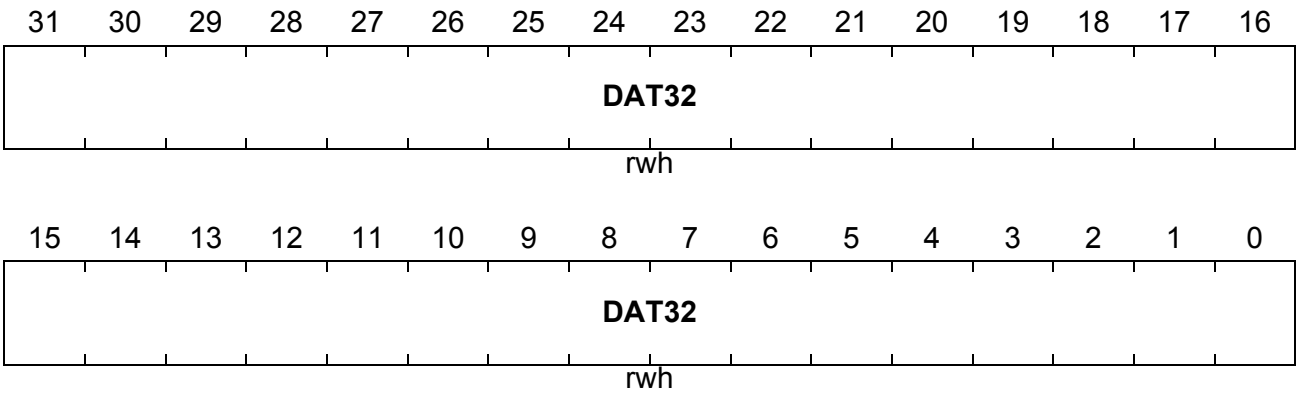
The DATA32/DATA16/DATA8 register is the single address interface to exchange IN and OUT data between the application software and the Endpoint buffers within the USB.

If the CPU performs a read/write operation to an Endpoint buffer, the selected Endpoint must not be changed until a whole packet is read/written. If another Endpoint is selected before the end of the packet is reached, data may be lost.

If the Endpoint is to be changed and if less than the defined packet length is written to the FIFO at the end of an IN transfer (CPU writes data), the CPU must set the FCON.DONE bit to 1.

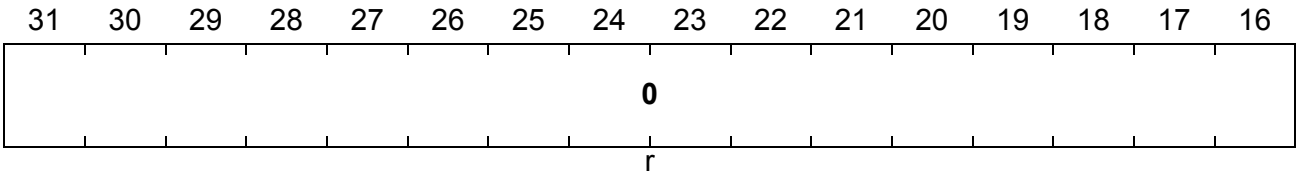
For the selected Endpoint, DATA32/ DATA16/DATA8 will be either read or written.

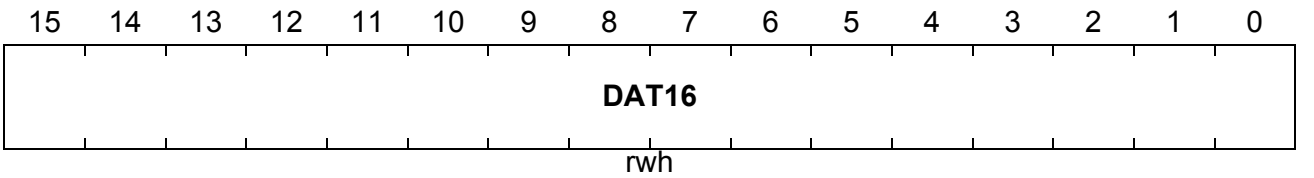
DATA32
Data Register for 32-bit Accesses **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DAT32	[31:0]	rwh	**DATA32 Register** OUT The data register contains the current data to be read by the CPU. IN The data register is used by the CPU to write data to be sent via USB to the selected Endpoint Buffer.

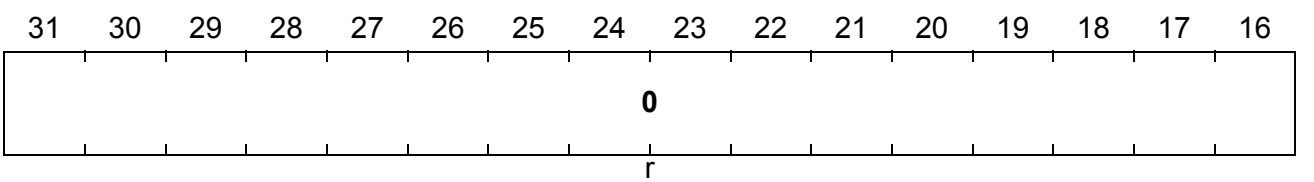
DATA16
Data Register for 16-bit Accesses **Reset Value: 0000 0000$_H$**

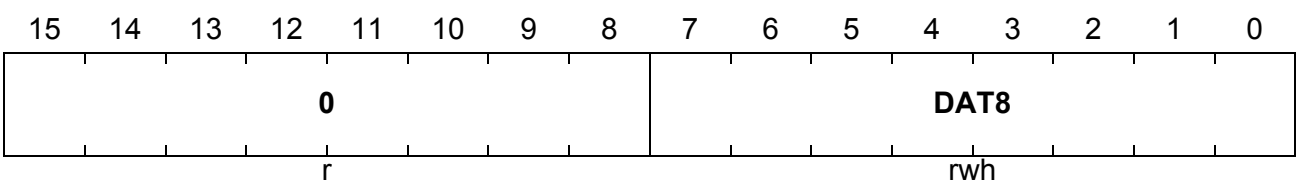




Field	Bits	Type	Description
DAT16	[15:0]	rwh	**DATA16 Register** OUT The data register contains the current data to be read by the CPU. IN The data register is used by the CPU to write data to be sent via USB to the selected Endpoint Buffer.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

DATA8
Data Register for 8-bit Accesses **Reset Value: 0000 0000$_H$**





Field	Bits	Type	Description
DAT8	[7:0]	rwh	**DATA8 Register** OUT The data register contains the current data to be read by the CPU. IN The data register is used by the CPU to write data to be sent via USB to the selected Endpoint Buffer.
0	[31:8]	r	**Reserved;** read as 0; should be written with 0.

26.2.5 Device Interrupt Registers

A flexible and sophisticated Interrupt Generation Unit is implemented, which minimizes the CPU load during USB data transfers.

As only one Endpoint at a time may cause an interrupt, the Endpoint interrupts may be mapped on a small set of interrupt lines using Interrupt Node Pointers.

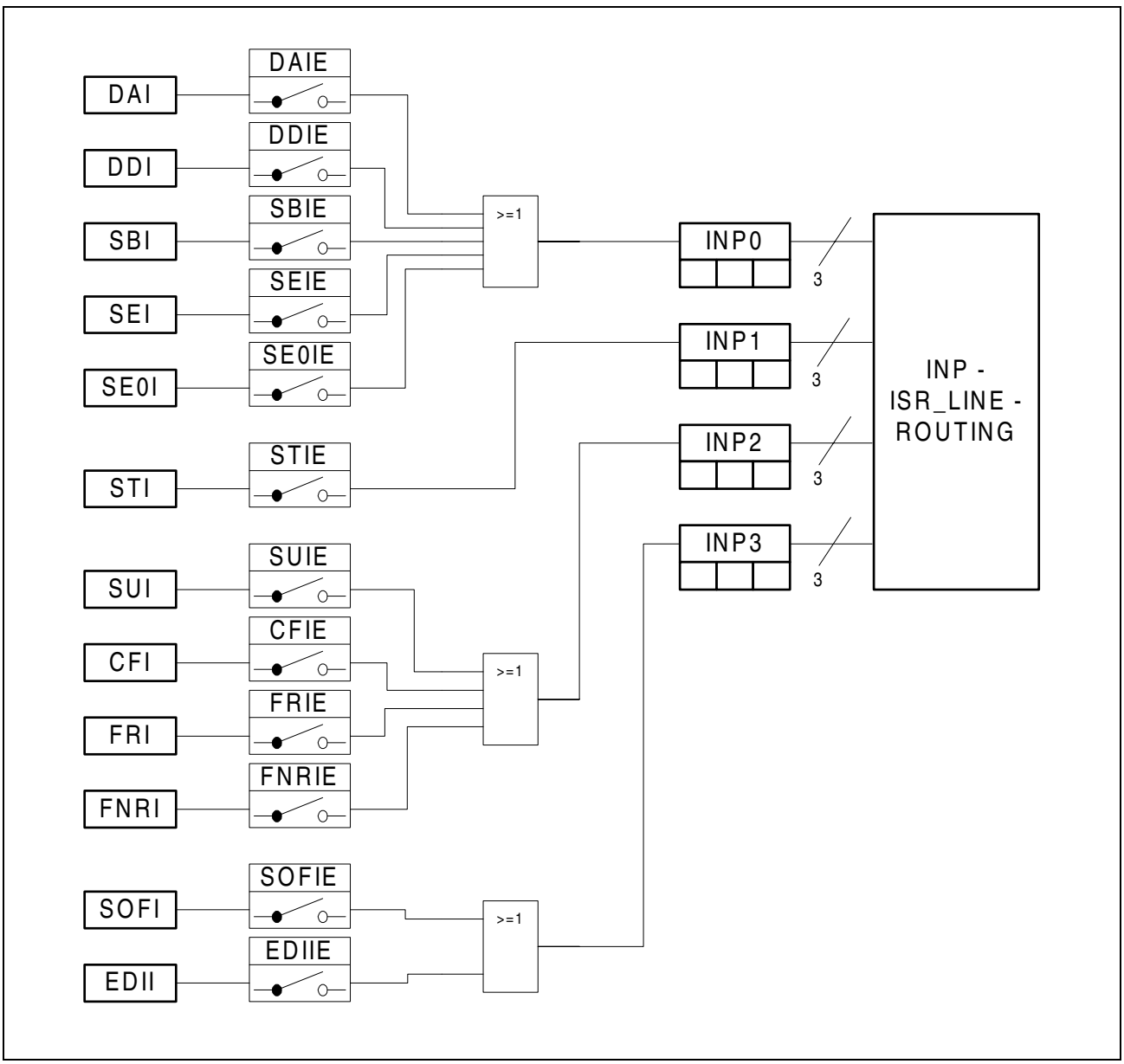


Figure 26-14 Device Interrupts

The DINP register contains the node vectors and node enable bits for each device interrupt node. The node vectors select the corresponding interrupt service request line.

DINP
Device Interrupt Node Pointer Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**INP 32**	**INP 31**	**INP 30**	**0**	**INP 22**	**INP 21**	**INP 20**	**0**	**INP 12**	**INP 11**	**INP 10**	**0**	**INP 02**	**INP 01**	**INP 00**
r	rw	rw	rw	r	rw	rw	rw	r	rw	rw	rw	r	rw	rw	rw

Field	Bits	Type	Description
INPn[2:0] **(n = 0 … 3)**	[2:0], [6:4], [10:8], [14:12]	rw	**Interrupt Node Pointer n** 000 Interrupt Service Request Line 0 is selected 001 Interrupt Service Request Line 1 is selected 010 Interrupt Service Request Line 2 is selected 011 Interrupt Service Request Line 3 is selected 100 Interrupt Service Request Line 4 is selected 101 Interrupt Service Request Line 5 is selected 110 Interrupt Service Request Line 6 is selected 111 Interrupt Service Request Line 7 is selected for Interrupt Node n.
0	3, 7, 11, [31:15]	r	**Reserved;** read as 0; should be written with 0.

The DIER register contains the enable bits for the different types of device interrupts. With these bits, the device interrupts can be individually enabled or disabled.

DIER
Device Interrupt Enable Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	FNRIE	FRIE	EDIIE	SOFIE	CFIE	SUIE	STIE	0			SE0IE	SEIE	SBIE	DDIE	DAIE
r	rw	rw	rw	rw	rw	rw	rw	r			rw	rw	rw	rw	rw

Field	Bits	Type	Description
DAIE	0	rw	**Device Attached Interrupt Enable** Setting bit DAIE enables the generation of a device interrupt when it is attached to the USB bus. 0 The device attached interrupt is disabled 1 The device attached interrupt is enabled
DDIE	1	rw	**Device Detached Interrupt Enable** Setting bit DDIE enables the generation of a device interrupt when it is detached from the USB bus. 0 The device detached interrupt is disabled 1 The device detached interrupt is enabled
SBIE	2	rw	**Suspend Begin Interrupt Enable** Setting bit SBIE enables the generation of a device interrupt if bit DIRR.SBI is set, this means the suspend mode is entered. 0 The suspend begin interrupt is disabled 1 The suspend begin interrupt is enabled
SEIE	3	rw	**Suspend End Interrupt Enable** Setting bit SEIE enables the generation of a device interrupt if bit DIRR.SEI is set, this means the suspend mode is left. 0 The suspend end interrupt is disabled 1 The suspend end interrupt is enabled

Field	Bits	Type	Description
SE0IE	4	rw	**Single Ended Zero Interrupt Enable** Setting bit SE0IE enables the generation of a device interrupt each time a single ended zero is detected for more than 2.5 µs (reset by host, not EOP). 0 The single ended zero interrupt is disabled 1 The single ended zero interrupt is enabled
STIE	8	rw	**Status Interrupt Enable** Bit STIE enables the generation of a device interrupt at the end of the status phase of a control transfer. 0 The status interrupt is disabled 1 The status interrupt is enabled
SUIE	9	rw	**Setup Interrupt Enable** Bit SUIE enables the generation of a device interrupt on a successful reception of a setup packet which must be processed by the CPU. 0 The setup interrupt is disabled 1 The setup interrupt is enabled
CFIE	10	rw	**Configuration Interrupt Enable** Bit CFIE enables the generation of a device interrupt on a successful reception of a new configuration from the host. 0 The configuration interrupt is disabled 1 The configuration interrupt is enabled
SOFIE	11	rw	**Start of Frame Interrupt Enable** Bit SOFIE enables the generation of a device interrupt on the detection of a start of frame packet on the USB. 0 The start of frame interrupt is disabled 1 The start of frame interrupt is enabled
EDIIE	12	rw	**End of Device Initialization Interrupt Enable** Bit EDIIE enables the generation of a device interrupt on the end of the device initialization. 0 The end of device initialization interrupt is disabled 1 The end of device initialization interrupt is enabled

Field	Bits	Type	Description
FRIE	13	rw	**FIFO Ready Interrupt Enable** Bit FRIE enables the FIFO ready Interrupt. 0 The FIFO ready interrupt is disabled 1 The FIFO ready interrupt is enabled
FNRIE	14	rw	**FIFO not Ready Interrupt Enable** Bit FNRIE enables the FIFO not ready Interrupt. 0 The FIFO not ready interrupt is disabled 1 The FIFO not ready interrupt is enabled
0	[7:5], [31:15]	r	**Reserved;** read as 0; should be written with 0.

The DIRR register contains the interrupt request flags of different types of device interrupts. The referring Interrupt request flag in DIRR is reset by writing the corresponding bit in DIRST. This should be done after DIRR has been read.

DIRR
Device Interrupt Request Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	FNRI	FRI	EDII	SOFI	CFI	SUI	STI	0			SE0I	SEI	SBI	DDI	DAI
r	rh	rh	rh	rh	rh	rh	rh	r			rh	rh	rh	rh	rh

Field	Bits	Type	Description
DAI	0	rh	**Device Attached Interrupt** Bit DAI is automatically set after detection of the USB device being attached to the USB bus.
DDI	1	rh	**Device Detached Interrupt** Bit DDI is automatically set after detection of the device being detached from the USB bus.
SBI	2	rh	**Suspend Begin Interrupt** Bit SBI is automatically set when the suspend mode is entered.

Field	Bits	Type	Description
SEI	3	rh	**Suspend End Interrupt** Bit SEI is automatically set when the suspend mode is left.
SE0I	4	rh	**Single Ended Zero Interrupt** Bit SE0I is set each time a single ended zero is detected for equal or greater than 2.5 µs. EOP (2-bit times) is not detected.
STI	8	rh	**Status Interrupt** Bit STI is set if the host requests a status transfer and the device answers with NACK (if bit EPVLD.EPV0 is 1 and EPDIR.DIR0 is set correctly, the device answers with ACK and STI is not set).
SUI	9	rh	**Setup Interrupt** Bit SUI is automatically set after a successful reception of a setup packet, which is not handled by the USB module and must be forwarded to the CPU. The setup packet itself is limited to 8 bytes and is stored at memory location 0.
CFI	10	rh	**Configuration Interrupt** Bit CFI is automatically set after a successful reception of a new configuration from the host for: • Configuration Value • Interface • Alternate Setting *Note: The new setting may be read from register CNFR.*
SOFI	11	rh	**Start of Frame Interrupt** Bit SOFI is automatically set after detection of a start of frame packet on the USB.
EDII	12	rh	**End of Device Initialization Interrupt** Bit EDII is automatically set after detection of an end of device initialization.
FRI	13	rh	**FIFO Ready Interrupt** Bit FRI is automatically set if the CPU selected an Endpoint (CEPS) and the bit FCON.FVLD indicates that the FIFO is ready for data transfer.

Field	Bits	Type	Description
FNRI	14	rh	**FIFO not Ready Interrupt** Bit FNRI is automatically set if the CPU requires a data transfer (accessing DATA32/16/8) while bit FCON.FVLD indicates that the FIFO is not ready.
0	[7:5], [31:15]	r	**Reserved;** read as 0; should be written with 0.

The DIRST register contains the interrupt reset bits for different device interrupts. The referring Interrupt request flag in DIRR is reset by writing the corresponding bit in DIRST. This should be done after DIRR has been read.

DIRST
Device Interrupt Reset Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	FNRR	FRR	EDIIR	SOFIR	CFIR	SUIR	STIR	0			SE0IR	SEIR	SBIR	DDIR	DAIR
r	w	w	w	w	w	w	w	r			w	w	w	w	w

Field	Bits	Type	Description
DAIR	0	w	**Device Attached Interrupt Reset** 0 No action is taken 1 Bit DIRR.DAI is reset
DDIR	1	w	**Device Detached Interrupt Reset** 0 No action is taken 1 Bit DIRR.DDI is reset
SBIR	2	w	**Suspend Begin Interrupt Reset** 0 No action is taken 1 Bit DIRR.SBI is reset
SEIR	3	w	**Suspend End Interrupt Reset** 0 No action is taken 1 Bit DIRR.SEI is reset

Field	Bits	Type	Description
SE0IR	4	w	**Single Ended Zero Interrupt Reset** 0 No action is taken 1 Bit DIRR.SE0I is reset
STIR	8	w	**Status Interrupt Reset** 0 No action is taken 1 Bit DIRR.STI is reset
SUIR	9	w	**Setup Interrupt Reset** 0 No action is taken 1 Bit DIRR.SUI is reset
CFIR	10	w	**Configuration Interrupt Reset** 0 No action is taken 1 Bit DIRR.CFI is reset
SOFIR	11	w	**Start of Frame Interrupt Reset** 0 No action is taken 1 Bit DIRR.SOFI is reset
EDIIR	12	w	**End of Device Initialization Interrupt Reset** 0 No action is taken 1 Bit DIRR.EDII is reset
FRR	13	w	**FIFO Ready Interrupt Reset** 0 No action is taken 1 Bit DIRR. FRI is reset
FNRR	14	w	**FIFO not Ready Interrupt Reset** 0 No action is taken 1 Bit DIRR.FNRI is reset
0	[7:5], [31:15]	r	**Reserved;** read as 0; should be written with 0.

26.2.6 Endpoint Interrupt Registers

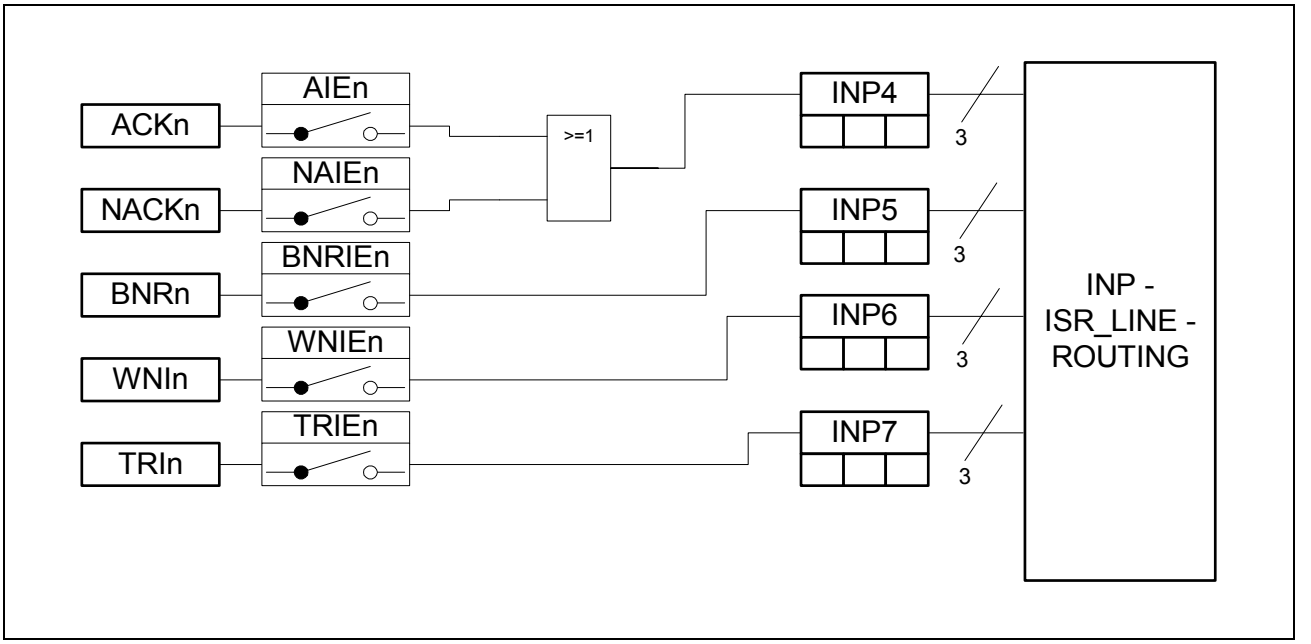


Figure 26-15 Endpoint Interrupts

The EPICn register is part of the Endpoint properties. With the Endpoint interrupt enable bits, the Endpoint specific interrupts can be individually enabled or disabled.

EPICn (n = 0 … 10)
Endpoint n Interrupt Control Register **Reset Value: XXXX XXXX$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**INP7 2n**	**INP7 1n**	**INP7 0n**	**0**	**INP6 2n**	**INP6 1n**	**INP6 0n**	**0**	**INP5 2n**	**INP5 1n**	**INP5 0n**	**0**	**INP4 2n**	**INP4 1n**	**INP4 0n**
r	rw	rw	rw	r	rw	rw	rw	r	rw	rw	rw	r	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											**TR IEn**	**WNI En**	**BNR En**	**NAC KEn**	**ACK En**
r											rw	rw	rw	rw	rw

Field	Bits	Type	Description
ACKEn	0	rw	**USB Acknowledge Interrupt Enable** Bit ACKEn enables the generation of an Endpoint specific acknowledge interrupt when bit EPIRn.ACKn is set. 0 The USB acknowledge interrupt is disabled 1 The USB acknowledge interrupt is enabled
NACKEn	1	rw	**USB not Acknowledged Interrupt Enable** Bit NACKEn enables the generation of an Endpoint specific not acknowledged interrupt when bit EPIRn.NACKn is set. 0 The USB not acknowledged interrupt is disabled 1 The USB not acknowledged interrupt is enabled
BNREn	2	rw	**Buffer not Ready Interrupt Enable** Bit BNREn enables the generation of an Endpoint specific interrupt when bit EPIRn.BNRn is set. 0 The buffer not ready interrupt is disabled 1 The buffer not ready interrupt is enabled
WNIEn	3	rw	**Warning Level Reached Interrupt Enable** Bit WNIEn enables the generation of an Endpoint specific interrupt when bit EPIRn.WNIn is set. 0 The warning level interrupt is disabled 1 The warning level interrupt is enabled
TRIEn	4	rw	**Transfer End Interrupt Enable** Bit TRIEn enables the generation of an Endpoint specific interrupt when bit EPIRn.TRIn is set. 0 The transfer end interrupt is disabled 1 The transfer end interrupt is enabled

Field	Bits	Type	Description
INPx[2:0]n (x = 7 … 4)	[18:16], [22:20], [26:24], [30:28]	rw	**Interrupt Node Pointer n** The Endpoint Interrupt Node Pointers contains the node vectors for each Endpoint interrupt node. The node vectors select the corresponding interrupt service request line. 000 Interrupt Service Request Line 0 is selected 001 Interrupt Service Request Line 1 is selected 010 Interrupt Service Request Line 2 is selected 011 Interrupt Service Request Line 3 is selected 100 Interrupt Service Request Line 4 is selected 101 Interrupt Service Request Line 5 is selected 110 Interrupt Service Request Line 6 is selected 111 Interrupt Service Request Line 7 is selected for Interrupt Node xn.
0	[15:5], 19, 23, 27, 31	r	**Reserved;** read as 0; should be written with 0.

The EPIRn register contains the interrupt request flags of the Endpoint specific interrupts. The referring Interrupt request flag in EPIRn is reset by writing the corresponding bit in EPIRSTn. This should be done after EPIRn has been read.

There is one set of five bits for each Endpoint. Four Endpoint interrupt sets are collected in one EPIRn register. Their organization of the registers has the following scheme shown with the example of EPIRn:

The bits for Endpoint0 are located in EPIR0, Byte 0.

…

The bits for Endpoint15 are located in EPIR3, Byte 3.

EPIRn (n = 0 … 3)
Endpoint Interrupt Request Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			TRIn3	WNIn3	BNRn3	NACKn3	ACKn3	0			TRIn2	WNIn2	BNRn2	NACKn2	ACKn2
r			rh	rh	rh	rh	rh	r			rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			TRIn1	WNIn1	BNRn1	NACKn1	ACKn1	0			TRIn0	WNIn0	BNRn0	NACKn0	ACKn0
r			rh	rh	rh	rh	rh	r			rh	rh	rh	rh	rh

Field	Bits	Type	Description
ACKnm (m = 0 … 3)	0, 8, 16, 24	rh	**USB Acknowledge** Bit ACKnm is automatically set after a successful action on the USB.
NACKnm (m = 0 … 3)	1, 9, 17, 25	rh	**USB not Acknowledged** Bit NACKnm is automatically set for all unsuccessful actions on the USB.
BNRnm (m = 0 … 3)	2, 10, 18, 26	rh	**Buffer not Ready Interrupt** Bit BNRnm is automatically set if the USB requires a data transfer and the referring EPVLD.EPVn bit for the addressed Endpoint is reset. *Note: Together with the setting of BNR00, EPVLD.EPV0 will be reset automatically if the predicted direction (EPDIR.DIR0) does not match the requested direction from the Host.*
WNInm (m = 0 … 3)	3, 11, 19, 27	rh	**Warning Level Reached Interrupt** Bit WNInm is automatically set if the EPBC.NEBn reaches the warning level set in WRNn.
TRInm (m = 0 … 3)	4, 12, 20, 28	rh	**Transfer End Interrupt** Bit TRInm is automatically set after the occurrence of an acknowledged End of Transfer (EOT).
0	[7:5], [15:13], [23:21], [31:29]	r	**Reserved;** read as 0; should be written with 0.

The EPIRSTn register contains the interrupt reset bits for the Endpoint specific interrupts. The referring Interrupt request flag in EPIRn is reset by writing the corresponding bit in EPRSTn. This should be done after EPIRn has been read.

EPIRSTn (n = 0 … 3)
Endpoint Interrupt Reset Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			TRIR3	WNIRn3	BNRRn3	NACKRn3	ACKRn3	0			TRIRn2	WNIRn2	BNRRn2	NACKRn2	ACKRn2
r			w	w	w	w	w	r			w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			TRIRn1	WNIRn1	BNRRn1	NACKRn1	ACKRn1	0			TRIRn0	WNIRn0	BNRRn0	NACKRn0	ACKRn0
r			w	w	w	w	w	r			w	w	w	w	w

Field	Bits	Type	Description
ACKRnm (m = 0 … 3)	0, 8, 16, 24	w	**USB Acknowledge Interrupt Reset** 0 No action is taken 1 Bit ACKnm is reset
NACKRnm (m = 0 … 3)	1, 9, 17, 25	w	**USB not Acknowledged Interrupt Reset** 0 No action is taken 1 Bit NACKnm is reset
BNRRnm (m = 0 … 3)	2, 10, 18, 26	w	**Buffer not Ready Interrupt Reset** 0 No action is taken 1 Bit BNRnm is reset
WNIRnm (m = 0 … 3)	3, 11, 19, 27	w	**Warning Level Reached Interrupt Reset** i0 No action is taken 1 Bit WNInm is reset
TRIRnm (m = 0 … 3)	4, 12, 20, 28	w	**Transfer End Interrupt Reset** 0 No action is taken 1 Bit TRInm is reset
0	[7:5], [15:13], [23:21], [31:29]	r	**Reserved;** read as 0; should be written with 0.

26.2.7 Interrupt Node Pointer

The USB has 8 interrupt lines at its disposal. The interrupt node pointers determine the referring interrupt line for each interrupt. We note that each interrupt is hard wired with one particular interrupt node pointer. The interrupts on device level are connected to INP0 … INP3. The Interrupts on Endpoint level are connected to INP4 … INP7.

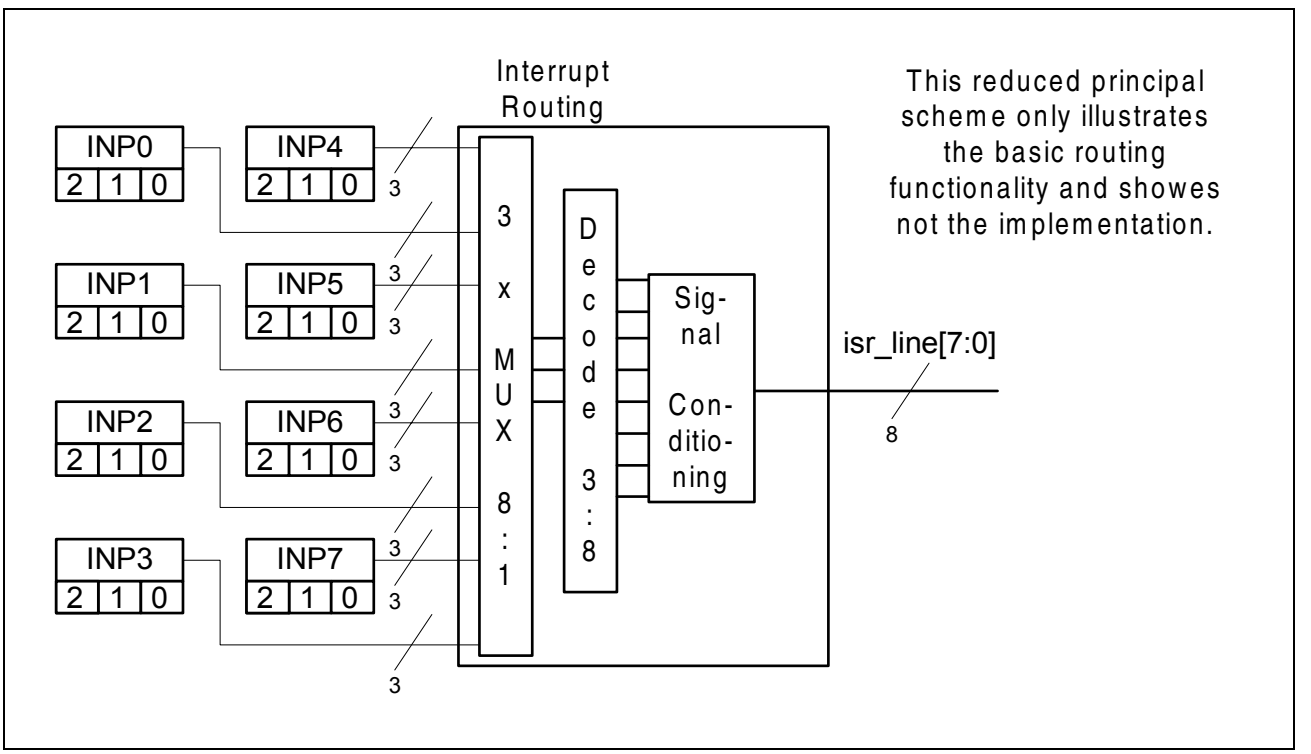


Figure 26-16 Interrupt Node Pointer

26.3 USB Module Implementation

This section describes USB module interfaces with the clock control, port connections, interrupt control, and address decoding.

26.3.1 Interfaces of the USB Modules

Figure 26-17 shows the TC1130 specific implementation details and interconnections of the USB module. The USB module is further supplied by clock control, interrupt control, address decoding, and port control logic. One DMA request can be generated by USB module.

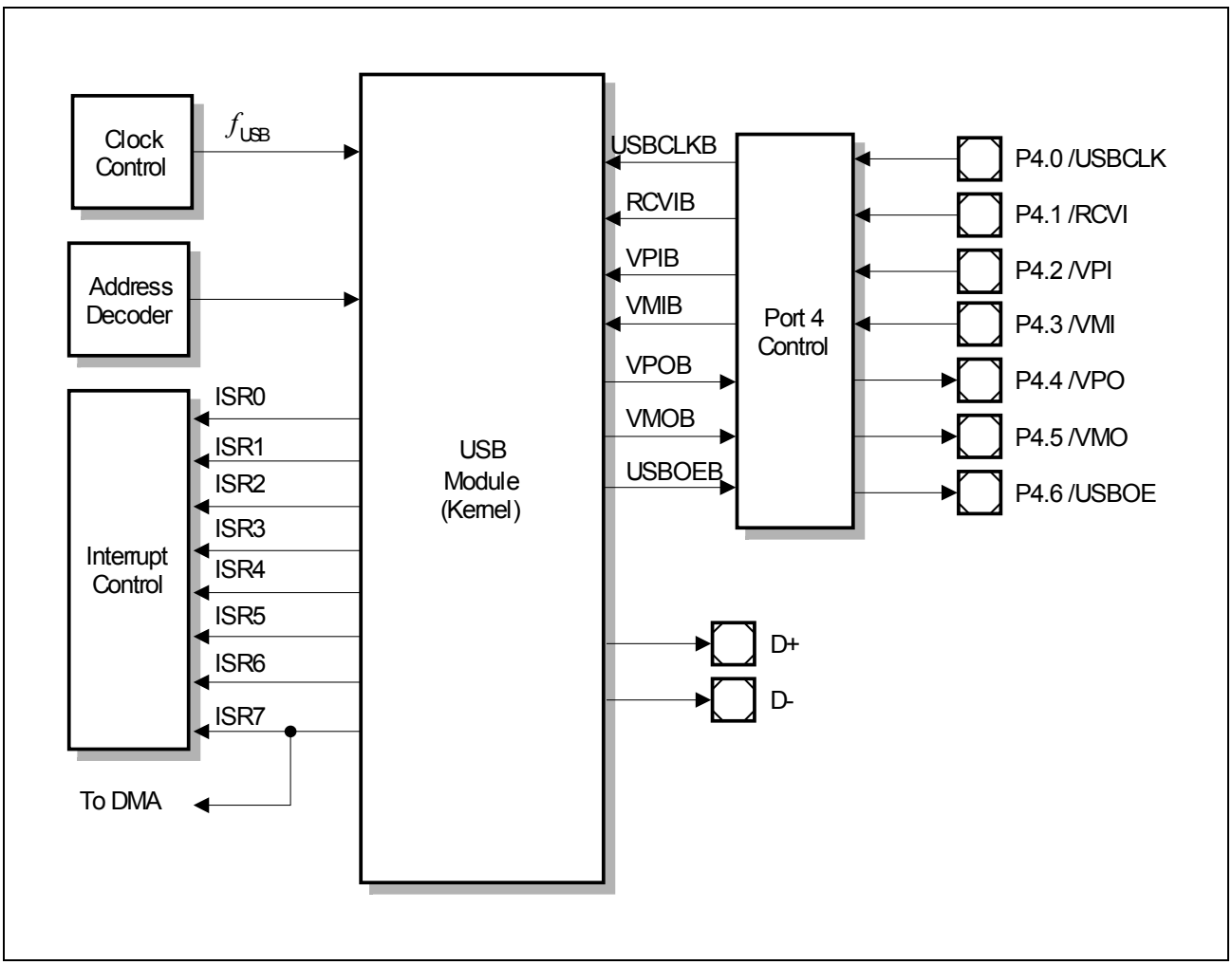


Figure 26-17 USB Module Implementation and Interconnections

26.3.2 USB Module Related External Registers

Figure 26-18 summarizes the module related external registers, which are required for USB programming (see also **Figure 26-13** for the module kernel specific registers).

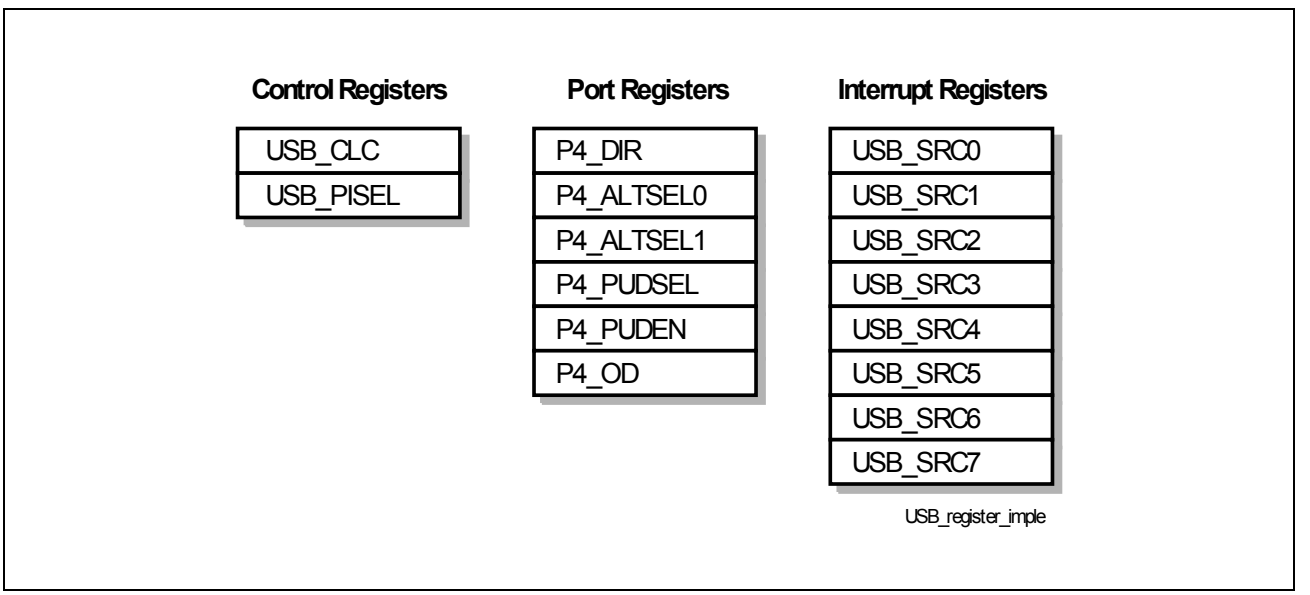
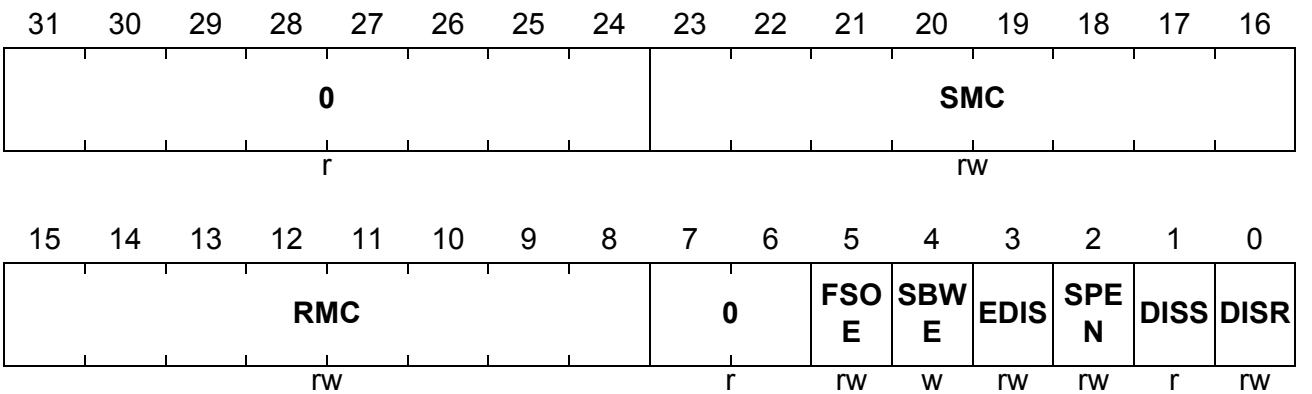


Figure 26-18 USB Implementation Specific Special Function Registers

26.3.2.1 Clock Control Registers

The clock control register allows the programmer to adapt the functionality and power consumption of an USB module to the requirements of the application. The table below shows the clock control register functionality, which is implemented for the USB module. USB_CLC is controlling the f_{USB} clock signal.

USB_CLC
USB Clock Control Register **Reset Value: 0000 0002$_H$**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the module.
SPEN	2	rw	**Module Suspend Enable for OCDS** Used for enabling the suspend mode.
EDIS	3	rw	**External Request Disable** Used for controlling the external clock disable request.
SBWE	4	w	**Module Suspend Bit Write Enable for OCDS** Defines whether SPEN and FSOE are write protected.
FSOE	5	rw	**Fast Switch Off Enable** Used for fast clock switch off in OCDS suspend mode.
RMC	[15:8]	rw	**8-Bit Clock Divider Value in RUN Mode**
SMC	[23:16]	rw	**Clock Divider Addition for Sleep Mode** Max. 8-bit adding value. *Note: FPI Bus only.*
0	[7:6] [31:24]	r	**Reserved;** read as 0; should be written with 0.

26.3.2.2 Peripheral Input Select Register

The USB module provides a Peripheral Input Select Register that is used to switch the input lines of the USB module kernels to either external transceiver (Port4) or internal transceiver. The register's definition is explained on Page 26-26.

26.3.2.3 Port Control

The interconnections between the USB modules and the port I/O lines are controlled in the port logics. The following port control operations selections must be executed (additionally to the PISEL programming):

- Input/output function selection (DIR register)
- Alternate function selection (ALTSEL0 and ALTSEL1 register)
- Input/Output driver characteristic control (PUDSEL, PUDEN and OD registers)

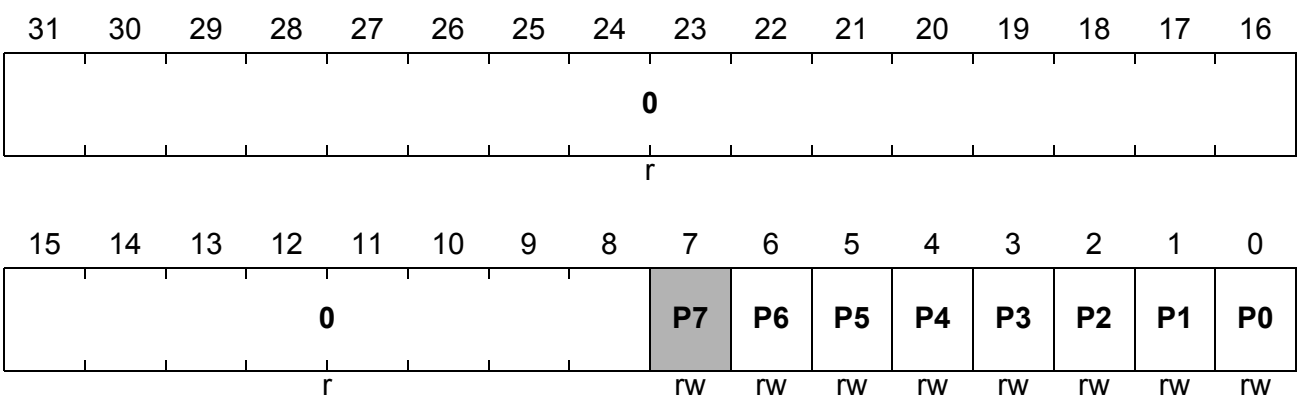
The USB port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as pull-up/down devices, port direction (input/output), open-drain, and alternate output selections. The I/O lines for the USB module are controlled by the port input/output control registers of Port 4.

Table 26-6 shows how bits and bit fields must be programmed for the required I/O functionality of the USB I/O lines.

Table 26-6 USB I/O Line Selection and Setup

Module	Port Lines	Input/Output Control Register Bits	I/O
USB	P4.0/USBCLK	P4_DIR.P0 = 0_B	Input
	P4.1/RCVI	P4_DIR.P1 = 0_B	Input
	P4.2/VPI	P4_DIR.P2 = 0_B	Input
	P4.3/VMI	P4_DIR.P3 = 0_B	Input
	P4.4/VPO	P4_DIR.P4 = 1_B	Output
		P4_ALTSEL0.P4 = 1_B	
		P4_ALTSEL1.P4 = 0_B	
	P4.5/VMO	P4_DIR.P5 = 1_B	Output
		P4_ALTSEL0.P5 = 1_B	
		P4_ALTSEL1.P5 = 0_B	
	P4.6/USBOE	P4_DIR.P6 = 1_B	Output
		P4_ALTSEL0.P6 = 1_B	
		P4_ALTSEL1.P6 = 1_B	

P4_DIR
Port 4 Direction Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
Pn **(n = 0-6)**	n	rw	**Port 4 Pin 0 - 6 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:8]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for USB I/O port control.

P4_ALTSELn (n = 0, 1)
Port 4 Alternate Select Register **Reset Value: 0000 0000$_H$**

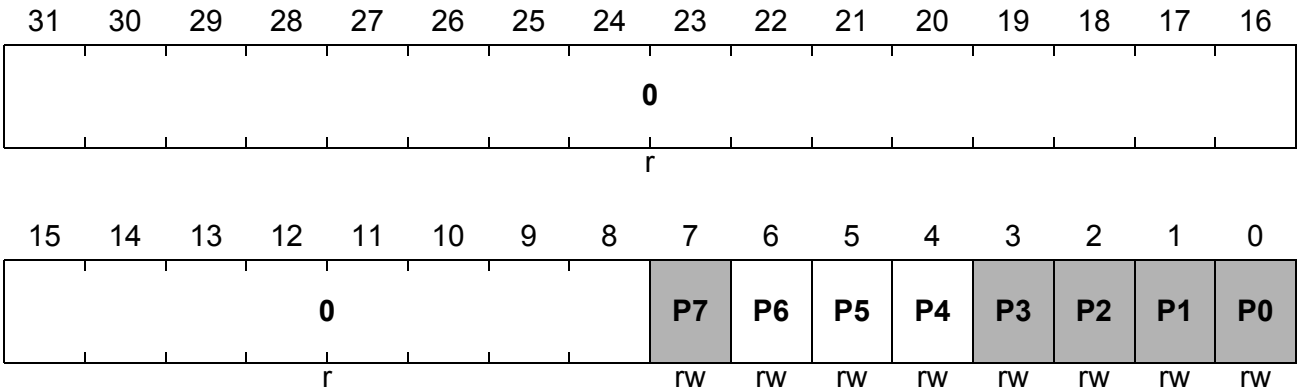


Table 26-7 Function of the Bits P4_ALTSEL0.Pn and P4_ALTSEL1.Pn (n = 4-6)[1)]

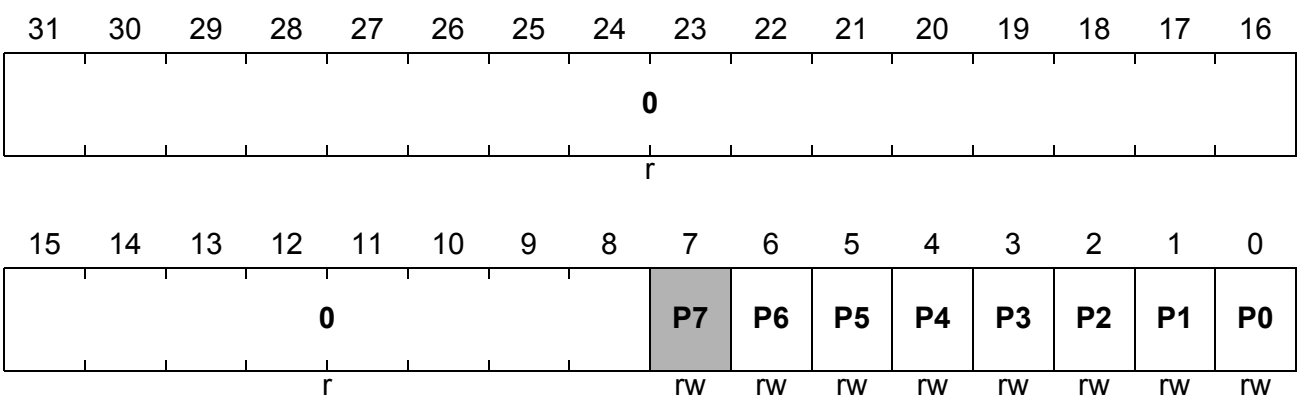
P4_ALTSEL0.Pn	P4_ALTSEL1.Pn	Function
1	0	Alternate Select 1

1) Shaded bits and bit field are don't care for USB I/O port control.

The USB ports also offer the possibility to configure the following output characteristics:

- Push/pull (optional pull-up/pull-down)
- Open drain with internal pull-up
- Open drain with external pull-up

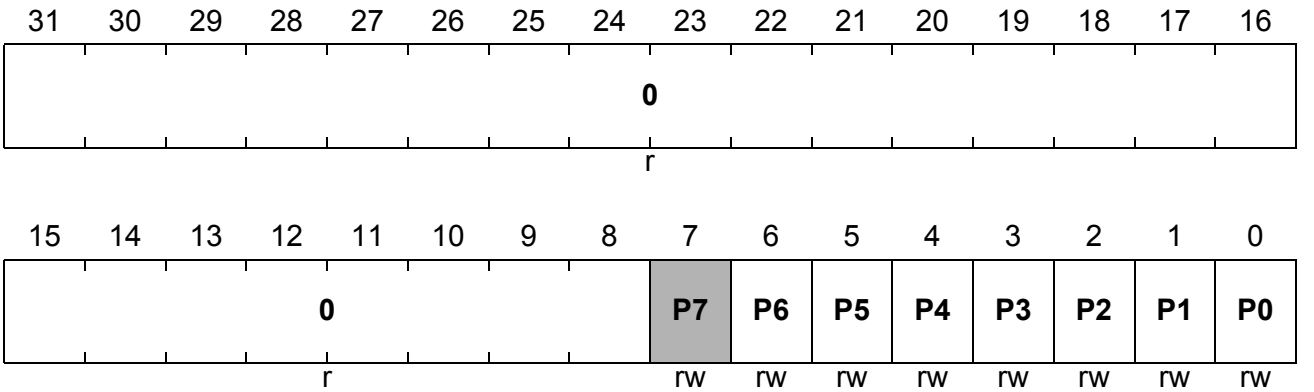
P4_PUDSEL
Port 4 Pull-Up/Pull-Down Select Register **Reset Value: 0000 00FF$_H$**



Field	Bits	Type	Description
Pn **(n = 0-6)**	n	rw	**Pull-Up/Pull-Down Select Port 4 Bit n**[1)] 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:8]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for USB I/O port control.

P4_PUDEN
Port 4 Pull-Up/Pull-Down Enable Register **Reset Value: 0000 00FF$_H$**

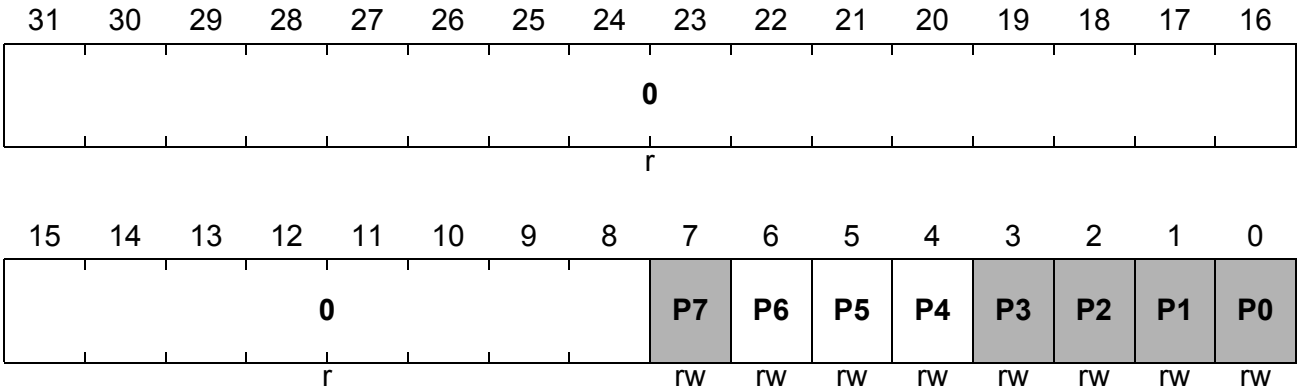


Field	Bits	Type	Description
Pn **(n = 0-6)**	n	rw	**Pull-Up/Pull-Down Enable at Port 4 Bit n**[1)] 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:8]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for USB I/O port control.

P4_OD
Port 4 Open Drain Control Register

Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
Pn **(n = 4-6)**	n	rw	**Port 4 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:8]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for USB I/O port control.

26.3.2.4 Interrupt Registers

The eight interrupts of the USB module are controlled by the following service request control registers:

USB_SRC[7:0]
USB Interrupt Service Request Control Register **Reset Values: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETR	CLRR	SRR	SRE	0	TOS	0		SRPN							
w	w	rh	rw	r	rw	r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number**
TOS	10	rw	**Type of Service Control**
SRE	12	rw	**Service Request Enable**
SRR	13	rh	**Service Request Flag**
CLRR	14	w	**Request Clear Bit**
SETR	15	w	**Request Set Bit**
0	[9:8], 11, [31:16]	r	**Reserved;** read as 0; should be written with 0.

Note: Further details on interrupt handling and processing are described in the chapter "Interrupt System" of the TC1130 System Units User's Manual.

26.3.3 DMA Requests

The DMA request output line of the USB module becomes active whenever its related interrupt line is activated. This line must be programmed properly. It is only valid when this interrupt line is triggered by WNI interrupt request via INP6 interrupt node pointer.

26.3.4 USB Register Address Ranges

In the TC1130, the registers of the USB module are located in the following address ranges:

- USB Registers: Module Base Address = F00E 2800$_H$
  Module End Address = F00E 28FF$_H$
- USB RAM Based Registers: Module Base Address = F00E 2000$_H$
  Module End Address = F00E 219F$_H$
- RAM: Module Base Address = F00E 21A0$_H$
  Module End Address = F000 27FF$_H$
- Absolute Register Address = Module Base Address + Offset Address
  (offset addresses see **Table 26-5**)

Note: The complete and detailed address map of the USB modules is described in the chapter “Register Overview” of the TC1130 System Units User’s Manual.

Note: As the real implementation of USB RAM Based Register in TC1130 is covered in the range F00E 2000$_H$ to F00E 20BF$_H$, the left area in USB RAM based register range F00E 20C0$_H$ to F00E 219F$_H$ can be used as USB RAM.

27 Micro Link Serial Bus Interface (MLI)

This chapter describes the two micro link serial bus interfaces MLI0 and MLI1 of the TC1130. It contains the following sections:

- Functional description of the MLI kernel, valid for MLI0 and MLI1 (see **Section 27.1**)
- MLI kernel register descriptions of all MLI kernel specific registers (see **Section 27.2**)
- TC1130 implementation specific details and registers of the MLI0/MLI1 modules (port connections and control, interrupt control, address decoding and clock control, see **Section 27.3**)

Note: All MLI kernel register names described in this section will be referenced in other parts of the TC1130 User's Manual with the module name prefix "MLI0_" for the MLI0 interface and by "MLI1_" for the MLI1 interface.

27.1 MLI Kernel Description

27.1.1 MLI Applications

- Data and program exchanging without intervention of CPU or PCP between microcontrollers of the AUDO - NG family.
- The internal architecture of the block allows the communication between controllers in different clock domains.
- Compatibility with the SSC interface.
- The read mode also allows a request to be made to the other controller for the desired data.
- Resources sharing. It is possible to use resources not available in the controller but present in another one.
- Capability of programming up to four different interrupts in the second controller by sending a command.

Figure 27-1 shows a general overview of the MLI location in the controller and its connection within another MLI.

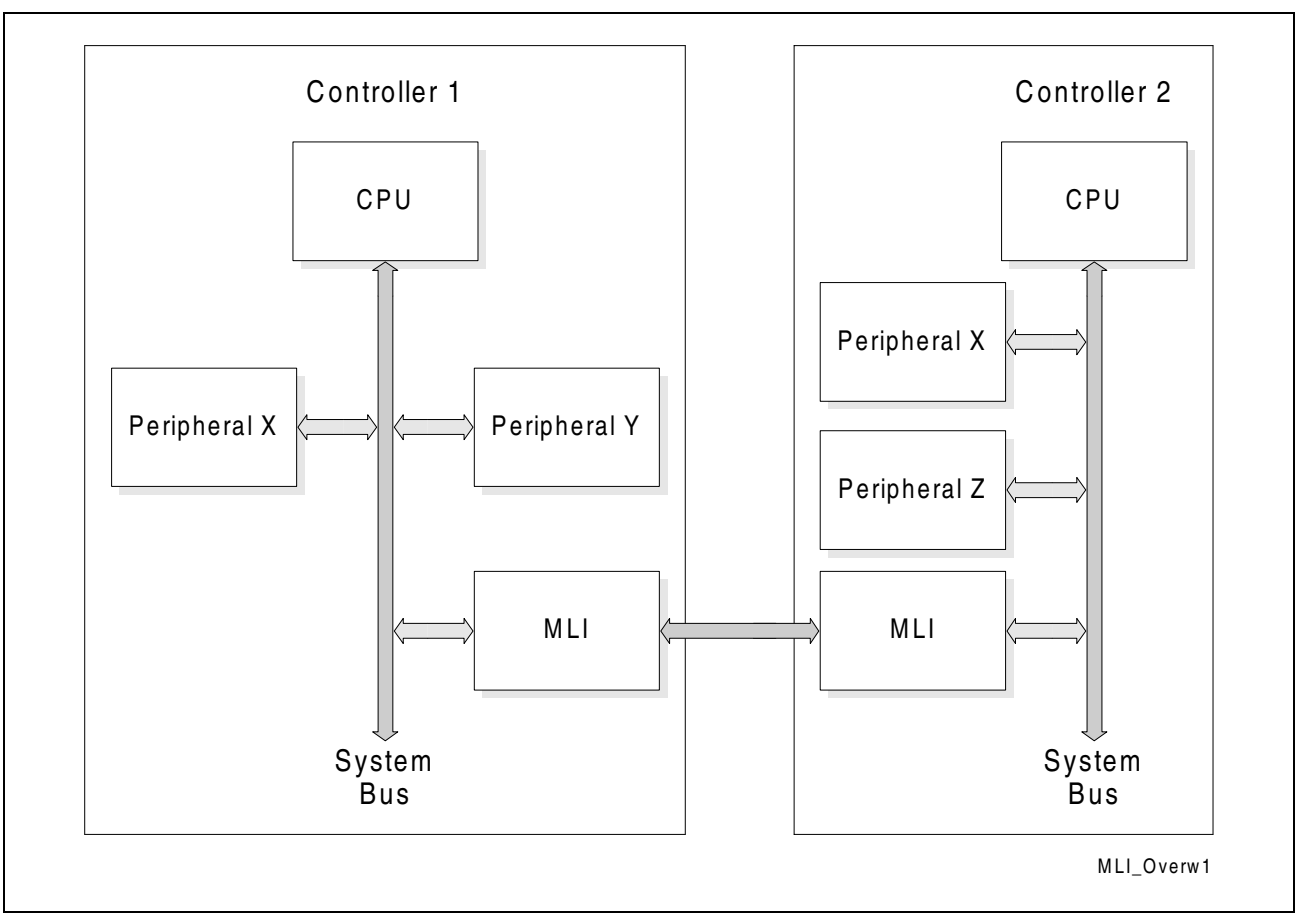


Figure 27-1 Location of MLI in the Controller and Connection

Figure 27-2 shows all the functional blocks of the MLI Module.

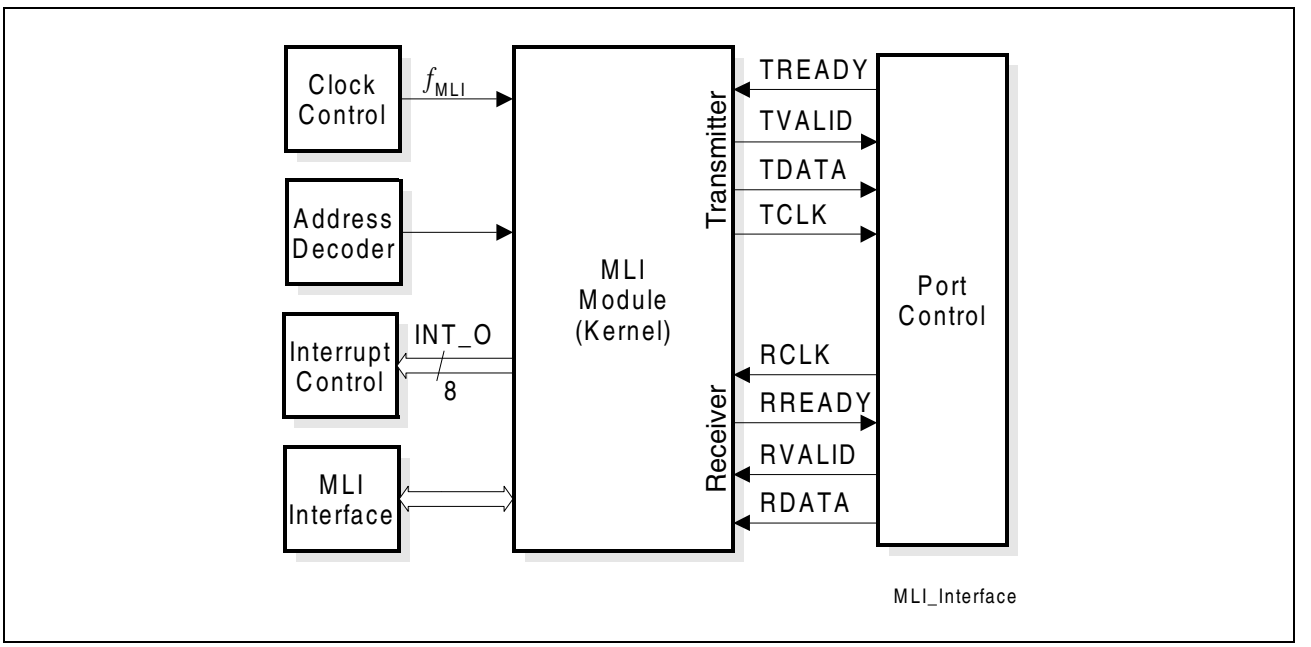


Figure 27-2 General Block Diagram of the MLI Module

27.1.2 Overview

The Micro Link Serial Bus Interface, referenced as MLI in the whole chapter, is dedicated for the serial communication between controllers of the AUDO - NG family. The communication is intended to be fast and intelligent due to an address translation system, and it is not necessary to have any special program in the second controller.

An overview of the MLI kernel is shown in **Figure 27-3**.

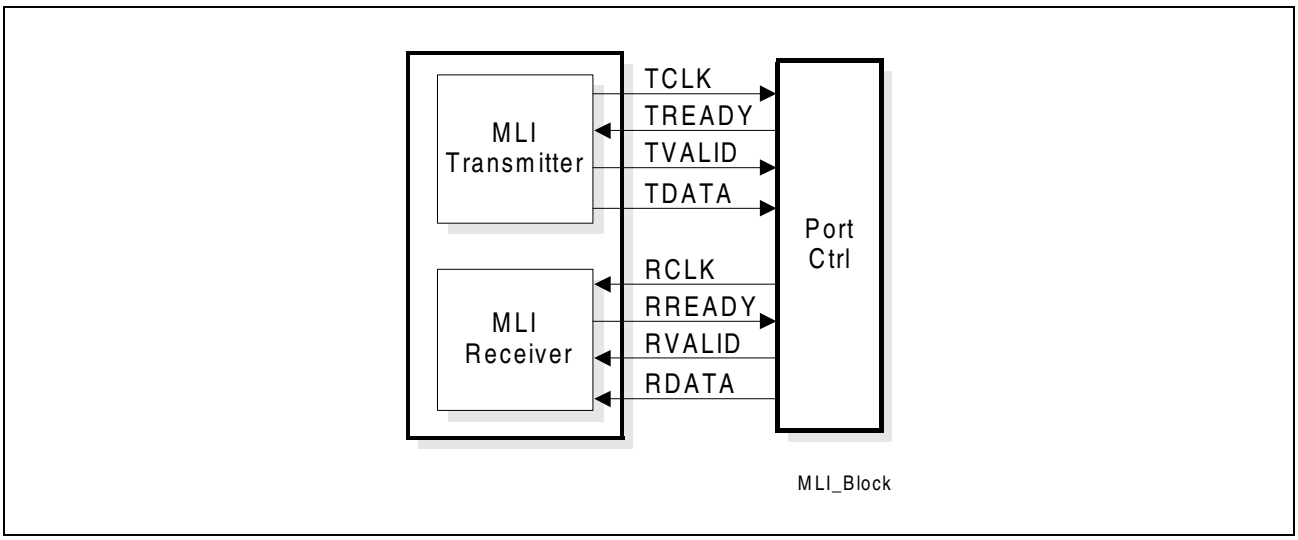


Figure 27-3 MLI Overview

Note: The prefixes T and R indicate if the corresponding signals belong to the MLI transmitter or to the MLI receiver.

Features

- Serial communication from the MLI transmitter to MLI receiver of another controller
- Module supports connection of each MLI with up to four MLI from other controllers
- Fully transparent read/write access supported (= remote programming)
- Complete address range of target controller available
- Special protocol to transfer data, address offset, or address offset and data
- Error control using a parity bit
- 32-bit, 16-bit, and 8-bit data transfers
- Address offset width: from 1 to 16 bits
- Baud rate: $f_{MLI}/2$ (symmetric shift clock approach),
  baud rate definition by the corresponding fractional divider

27.1.2.1 Naming Conventions

Local and Remote Controller

The names “local” and “remote” controller (device) are assigned to the two partners (microcontrollers with MLI modules) of a serial MLI connection. The controller with an MLI module that operates as a master of the serial MLI connection is defined as local controller. A local controller handles data operations with transfer windows and further initiates all control tasks (control, address, and data transmissions) that are required for the data transfer/request between local and remote controller.

The controller with an MLI module that operates as a slave of the serial MLI connection is defined as remote controller. A remote controller handles data operations with remote windows and executes the tasks that have been assigned/requested by the local controller.

Due to the full duplex operation capability of an MLI module, two serial MLI connections can be installed simultaneously (both transmitters can send a frame to their receivers). This means, each microcontroller with an MLI module is able to operate as local controller as well as remote controller at the same time.

Transfer Window

A transfer window is an address space in the address map of the local controller that is typically not assigned to memories or peripheral units. Transfer windows are always assigned to fixed address space (base address and size) in a specific microcontroller. Each MLI module supports up to four transfer windows with two different window sizes: four small transfer windows with 8 Kbytes and four large transfer windows with 64 Kbytes. Address and data information that has been written or read to/from transfer windows can be detected and handled by the MLI module of the local controller.

Remote Window

A remote window is an area in the address space of the remote controller. Remote window parameters (base address and window size) are defined and controlled by the local microcontroller. The size of a remote window is defined in a 4-bit coded buffer size parameter that defines the number of variable address bits of a remote window. Each MLI module supports up to four remote windows.

Pipe

A pipe defines the logical connection between an MLI module in the local controller and an MLI module in the remote controller. The logical connection of a pipe maps the transfer window in the local controller to its corresponding remote window in the remote controller. The MLI module supports four pipes.

27.1.2.2 MLI Communication Principles

The communication principle of the MLI modules allows that data is transferred between a local and a remote controller without intervention of a CPU. Data transfers are always triggered in the local controller by read or write operations to memory locations that are located in a transfer window of the local controller. All control tasks (control, address, and data transmissions) that are required for the data read/write accesses between local and remote controller are handled autonomously by the two MLI modules.

A **write access to a location within a transfer window** of the local controller is detected by the MLI transmitter. This detection initiates a transfer of the data that has been written to the transfer window from the local microcontroller to the MLI receiver of the remote controller which places the data at an address location in a remote window of the remote controller.

A **read access from a location of a transfer window** in the local controller returns dummy data and initiates the MLI connection to request data from the remote controller. Data is read in the remote controller by the MLI module from an address location within the remote window and transferred to the local controller where it is stored in its MLI receiver registers. Afterwards, the CPU in the local controller is informed by an interrupt that the requested data is now available and can be read from a register.

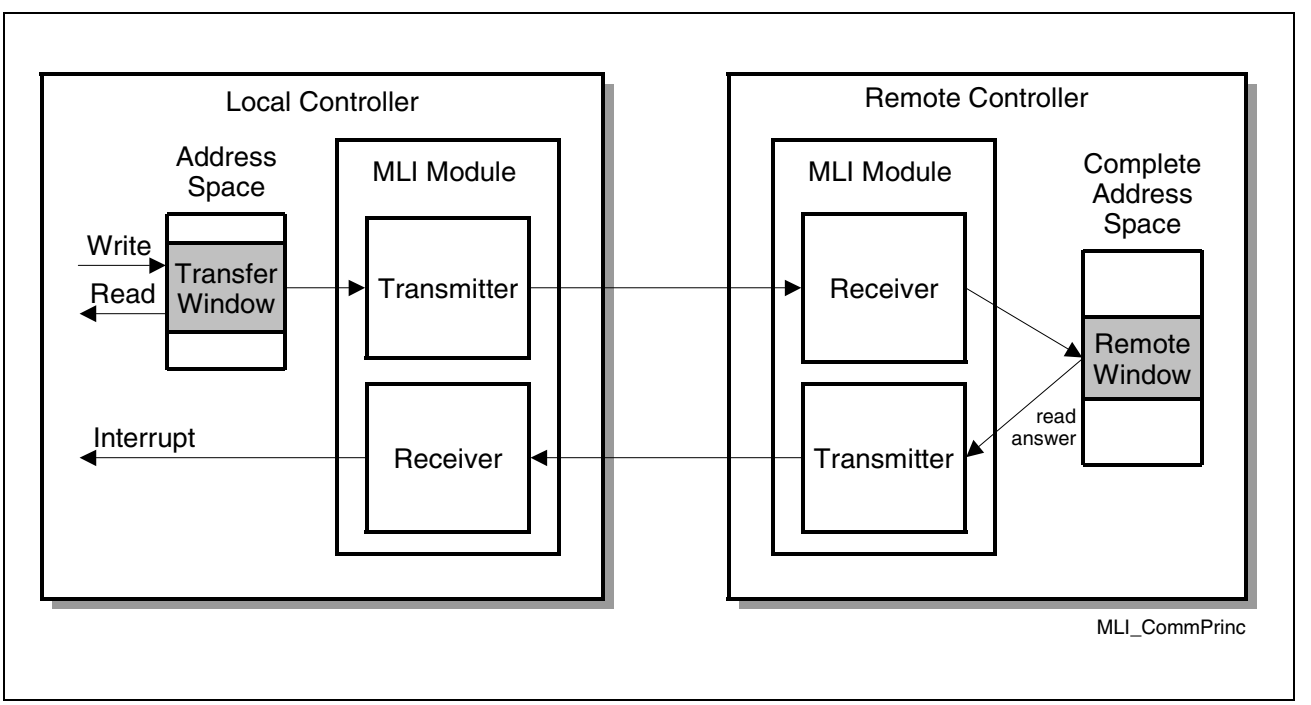


Figure 27-4 MLI Communication Principles

27.1.3 General Description

The communication between both controllers is based in an address translation table that allows the MLI transmitter from the first controller just sending an offset relative to these addresses instead of the full 32-bit address. The consistency for these addresses is guaranteed because the first controller sends all of them to the second controller and this one stores them as explained in **Section 27.1.7.5** and in **Section 27.1.7.6**. Each of the addresses defines a pipe and together with the Buffer Size parameter will define a buffer in the second controller address map. For each MLI transmitter there will be up to four pipes. A pipe may be seen as a logical connection between two controllers.

Figure 27-7 shows the organization in memory of the MLI transfer windows and their possible correspondence in the second controller address map.

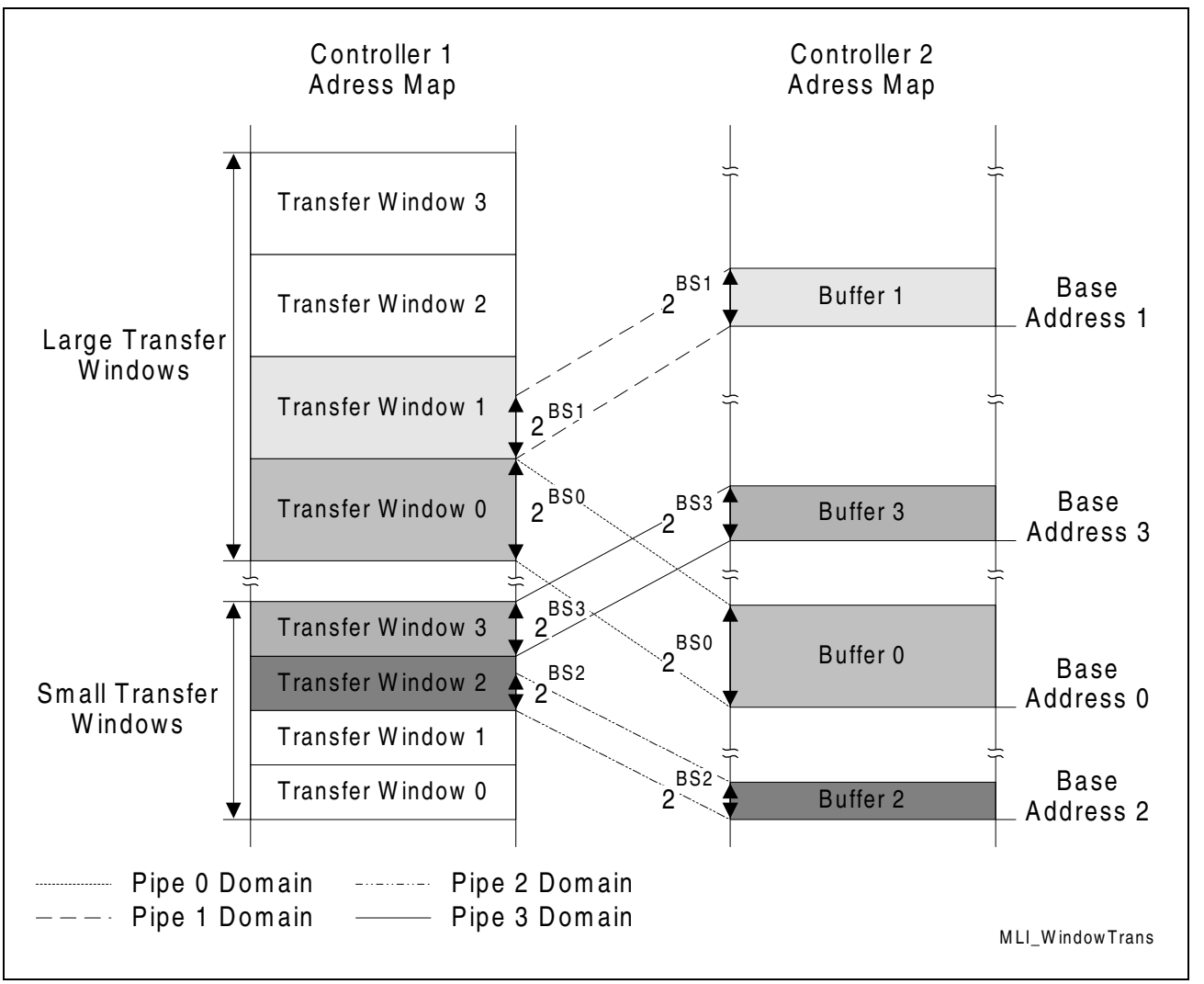


Figure 27-5 Transfer Window Base Address Copy

Note: BSx is the buffer size of each of the different pipes (where x = 0, 1, 2, 3).

Figure 27-6 illustrates a general overview of a two MLI connected, in which the transfer windows, TW, and the move engine of the receiver have been detailed.

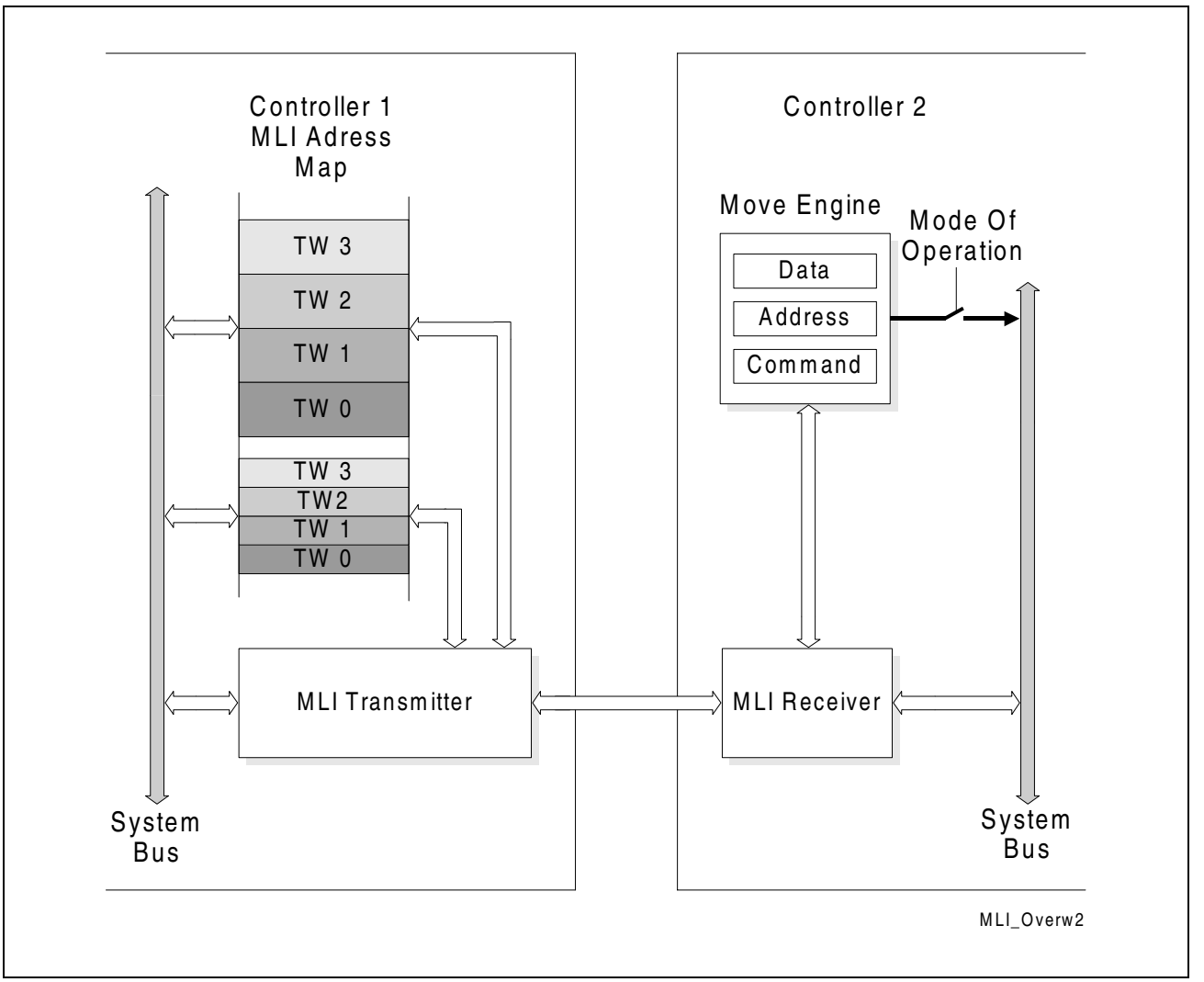


Figure 27-6 MLI Connection Overview

A **write access to a transfer window** in controller 1 leads to a transfer from the MLI transmitter to the MLI receiver on controller 2. The received information (incl. data and address or the command) is stored in the MLI receiver. There it is available for the CPU of controller 2 or the move can be executed autonomously by the MLI move engine (according to the selected access protection).

A **read access to a transfer window** delivers a dummy value on the system bus of controller 1. The read request is transferred to the MLI receiver on controller 2. If enabled, the MLI move engine executes the read operation autonomously and the requested data will be sent back to the MLI on controller 1 (by the MLI transmitter on controller 2 to the MLI receiver of controller 1). When this information is available in the MLI module of controller 1, an interrupt can be generated and the CPU (or a DMA, etc.) of controller 1 can read the requested data.

Figure 27-7 shows the process of transmission of the base address for each transfer window.

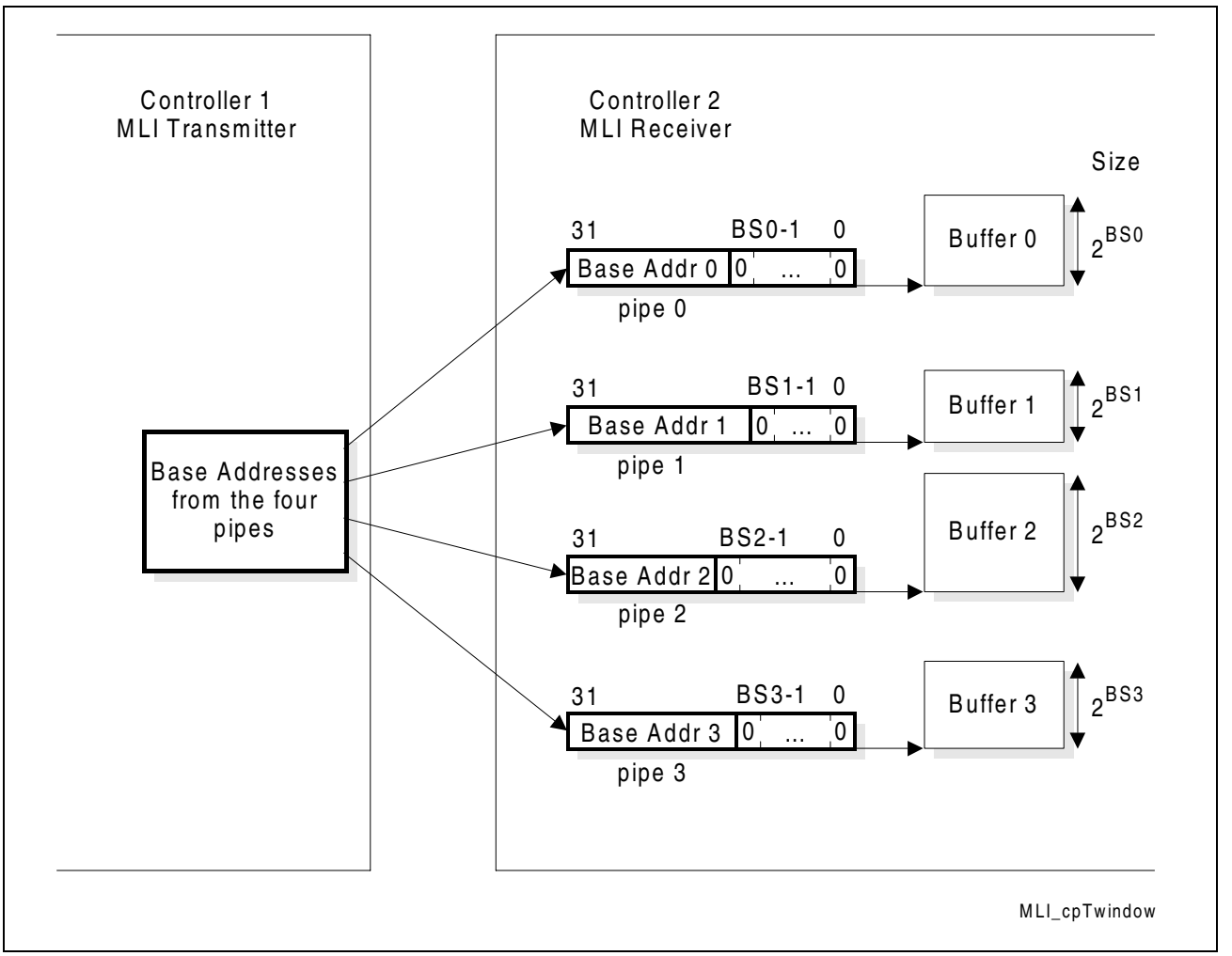


Figure 27-7 Transfer Window Base Address Copy

Note: BSx is the buffer size of each of the different pipes (where x = 0, 1, 2, 3). The selected buffer size must not exceed the size of the targeted transfer window (8 Kbytes for small transfer windows).

Within the offset (its width in bits is the same as indicated in the buffer size), the MLI transmitter from the first controller sends a reference to the pipe in use. When the MLI receiver obtained this data it will know what is the absolute address by simply concatenating the offset to the base address of the pipe.

Note: A pipe should always be accessed by either its small transfer window or the corresponding large transfer window. Mixing accesses via both window types to the same pipe is possible but not recommended (optimized frames require a single transfer window type). The used transfer window type of a pipe can be different from those of the other pipes.

Figure 27-8 illustrates the address translation process from the first controller to the second one.

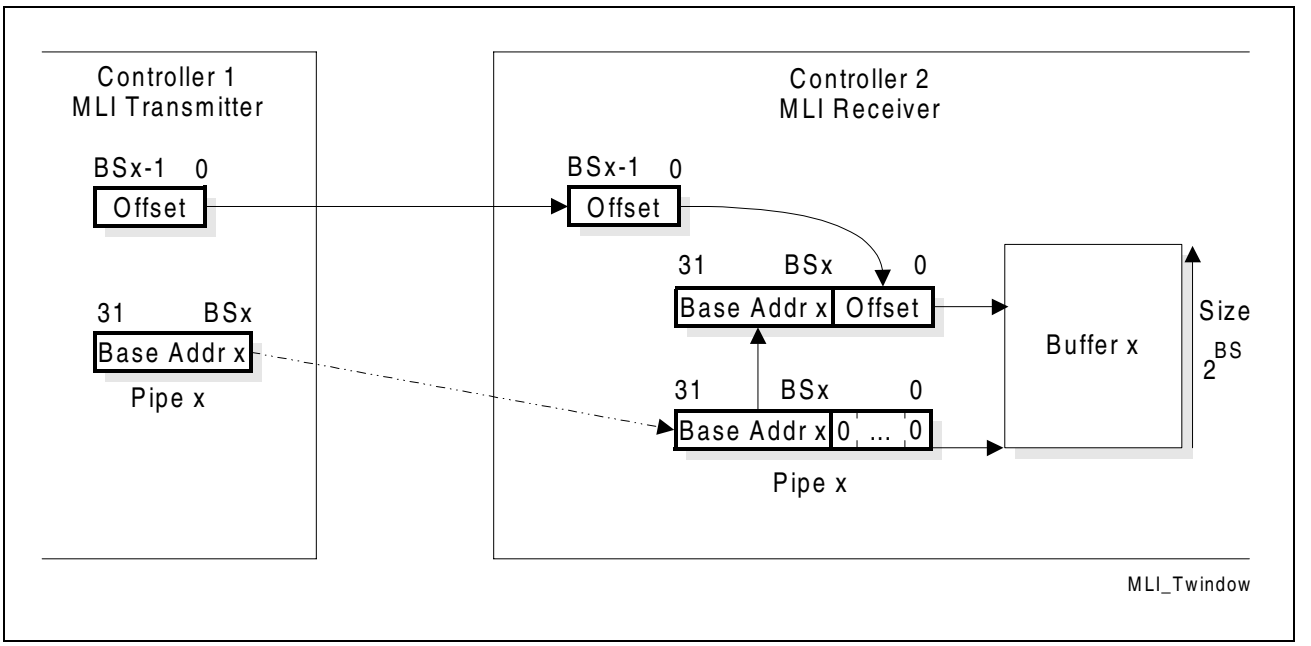


Figure 27-8 Address Translation Process

The kernel MLI includes an optimized mode to transfer data blocks. Whenever the MLI transmitter detects that the new address and the previous one follow a predictable scheme, it will send just the data reducing this way the bits to be transferred. This mode is based in a prediction method of the new address in the MLI receiver of the second controller. In fact the MLI receiver will automatically update the address, as explained in the next sections.

From the point of view of the connection, each MLI transmitter will have four READY, four VALID, one CLK and one DATA possible connections with the external world. This will provide the possibility of connecting each MLI transmitter with up to four MLI receivers in other controllers although the MLI transmitter will not have the possibility of transferring data to two or more different MLI receivers at the same time.

Each MLI receiver will have one READY, four VALID, four CLK and four DATA possible connections with the external world. With this scheme, each MLI receiver could be connected with up to four MLI transmitters in other controllers although the MLI receiver will not have the be possibility of receiving data from two or more different MLI transmitters at the same time.

Figure 27-9 illustrates an example of connection between two MLI.

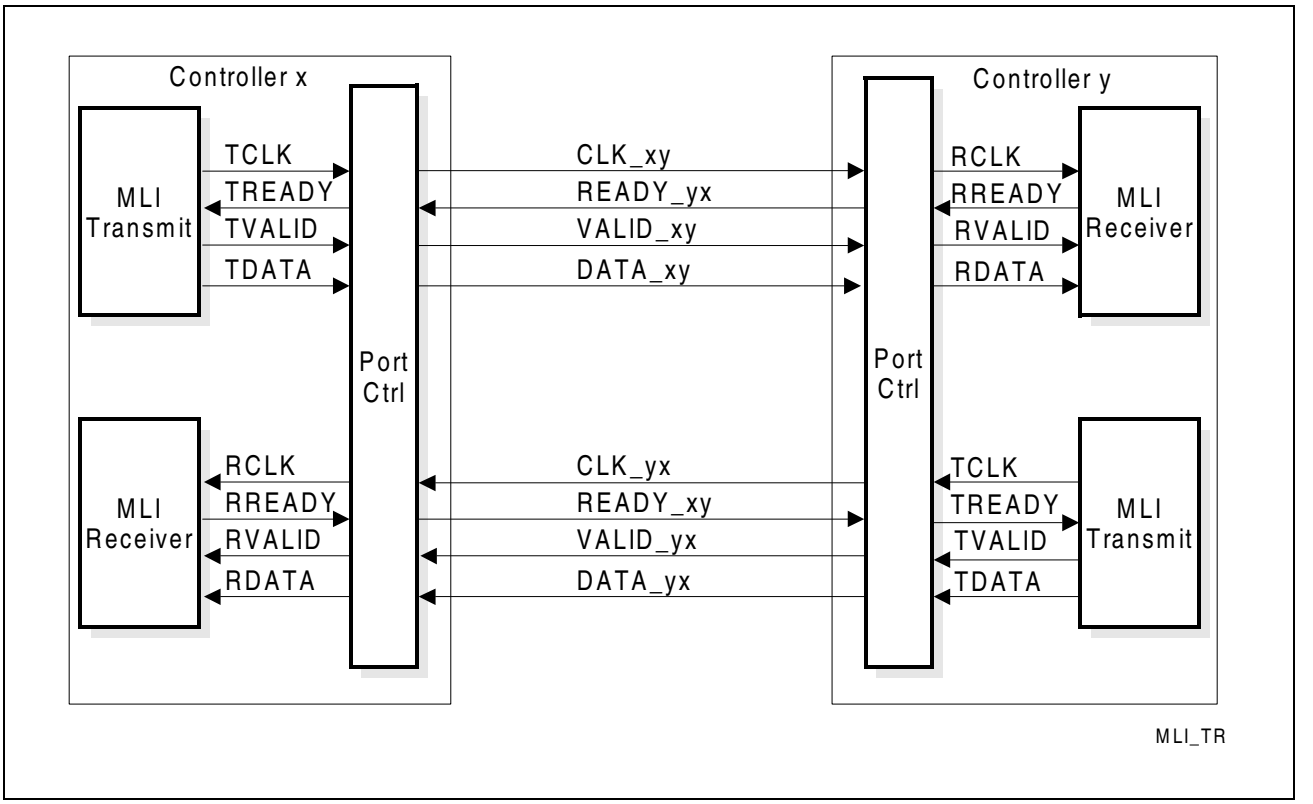


Figure 27-9 MLI Transmitter - Receiver Connection

Note: The suffixes x and y indicate the source and destination of the signals.

Figure 27-10 illustrates an example of connection between three different MLI.

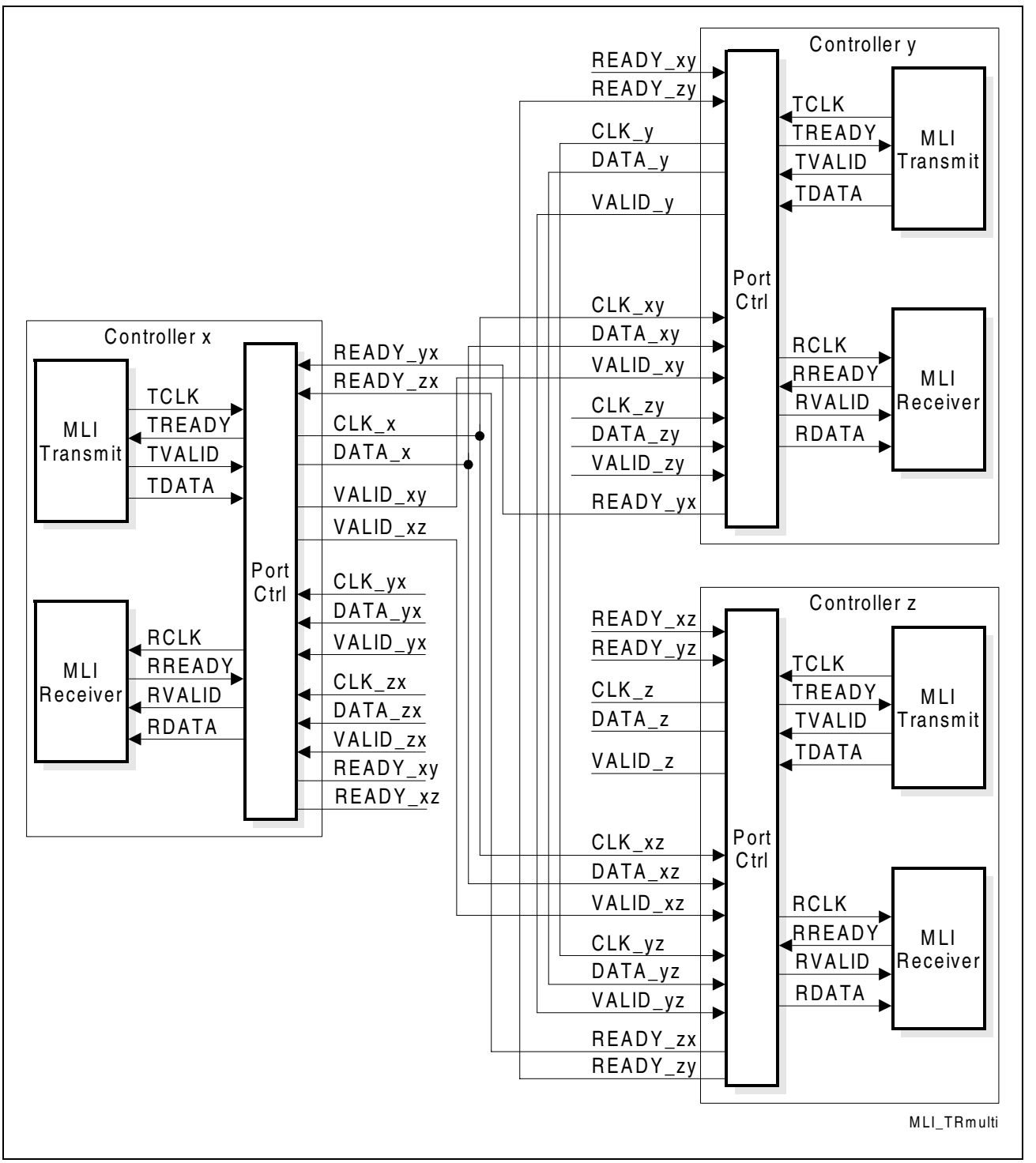


Figure 27-10 MLI Transmitter - Receiver Connection

Note: The suffixes x, y and z indicate the source and destination of the signals. For instance CLK_xz connects the clock signal CLK_x in the MLI transmitter x and finishes in the MLI receiver z. The signals selection is made by programming the OICR register.

27.1.4 Handshake Description

The transmission may start whenever the MLI receiver in the other controller is ready to receive data, i.e. READY signal is high. When the MLI transmitter from the first controller wants to start the transmission, it has to set the VALID signal high and it will hold it as long as it sends the data. When the VALID rising edge is detected by the MLI receiver in the other controller, it will set the READY signal to low level again.

When the transmission is finished, the MLI transmitter resets the VALID signal to zero and looks if the READY signal is low level again. This will indicate the MLI transmitter in the first controller that the MLI receiver has acknowledged the transmission. It will also reset the ready delay counter (TSTATR.RDC) and it will start counting TCLK clock periods. The counter will be stopped when the MLI transmitter detected that the READY signal is high level again or in the counting overflow.

If the MLI receiver sets the READY signal high again in a number of clock periods less than a programmed number (maximum delay for parity, TCR.MDP), this will indicate the MLI transmitter in the first controller that the MLI receiver is prepared for a new transmission and the previous one was received without parity error.

A detailed explanation about the handshake timing and clock domains in the MLI transmitter and the receiver may be found in **Section 27.1.11**.

The handshaking in a situation without error is shown in **Figure 27-11**.

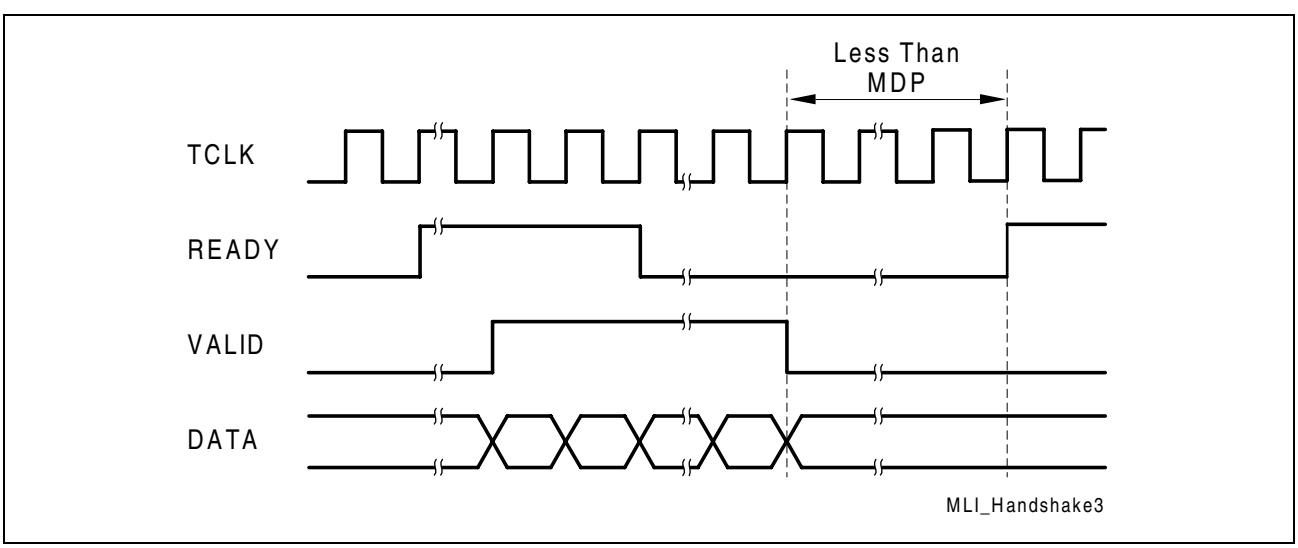


Figure 27-11 MLI Transmitter - Receiver Handshake in Transfer Without Error

Note: The signals are seen from the MLI transmitter in the first controller. When VALID is not asserted, the DATA line will have only a value (one or zero) depending on the programmed value in TCR.DNT and its chosen polarity.

The delay between the falling edge of VALID and the rising edge of READY is measured by the ready delay counter TSTATR.RDC. This value is compared to TCR.MDP on the transmitter side in order to detect when the receiver has signaled a parity error.

Figure 27-12 illustrates a situation of non acknowledge. The Ready signal remains high when the VALID is set low level again.

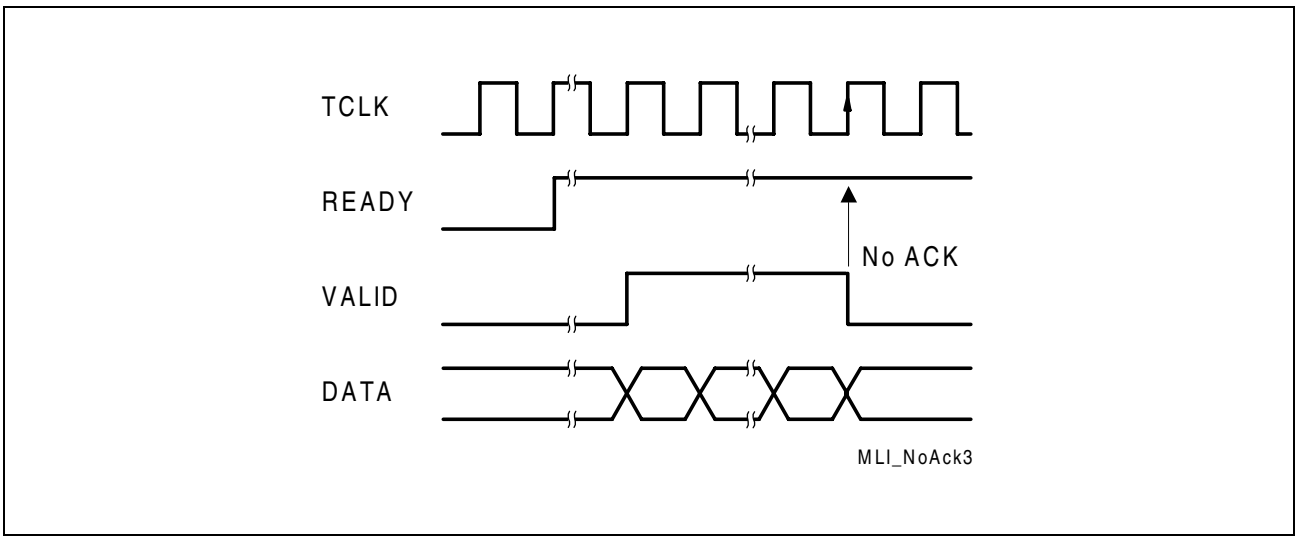


Figure 27-12 Non Acknowledge Situation

Note: The signals are seen from the MLI transmitter in the first controller.

If the READY signal raises after a number of TCLK clock periods (measured by TSTATR.RDC) is greater than the maximum delay for parity programmed value (TCR.MDP), then this situation will be interpreted as parity error, as shown in **Figure 27-13**.

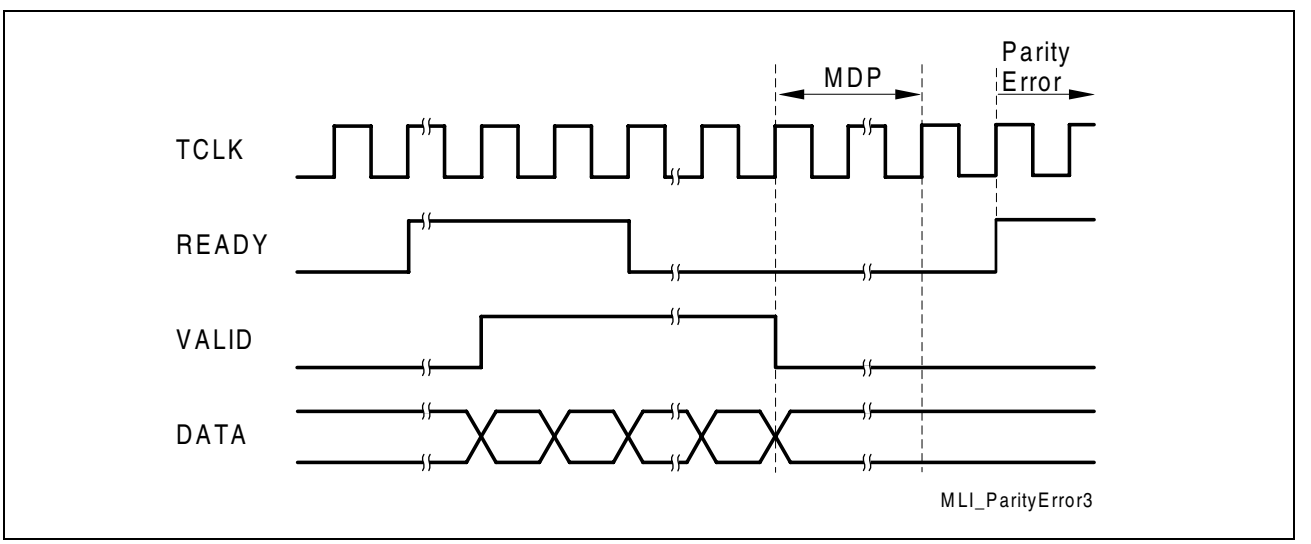


Figure 27-13 Parity Error Situation

Note: The signals are seen from the MLI transmitter in the first controller.

If a non acknowledge persists, the MLI transmitter will keep on counting the number of non acknowledge errors and if a maximum number is reached then a time-out error is produced. **Figure 27-14** illustrates this situation.

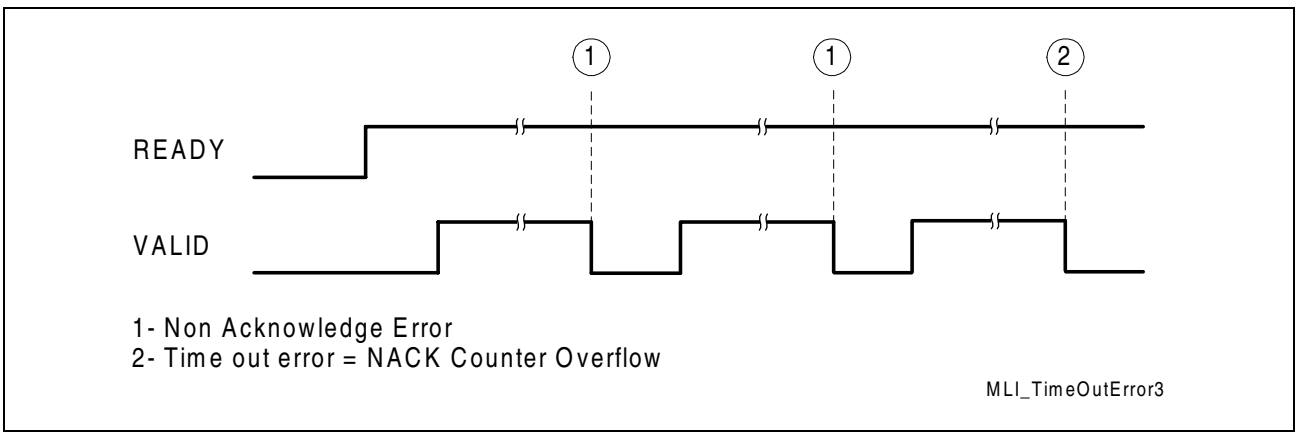


Figure 27-14 Time-out Error Situation

Note: The signals are seen from the MLI transmitter in the first controller.

All the error situations and the actuations taken by both parts are explained in **Section 27.1.7.8** and **Section 27.1.8.6**.

27.1.5 Startup Procedure

During the startup procedure of the MLI, an appropriate value for the maximum delay for parity signaling must be set up in the receiver and in the transmitter. Therefore, the following actions must be taken:

- The overall loop delay (total propagation delay between the transmitter and the receiver, including the output and the input driver delay, the line propagation and the synchronization time) must be measured.
- The appropriate MDP value must be programmed in the transmitter and then transferred to the receiver (automatic receiver setup by a command frame on pipe 1).
- Dummy frames with parity error and without parity error must be sent in order to check for a correct reply of the receiver.
- If the parity error signaling is working correctly, the setup is finished and normal frame traffic can be started.

For the measurement of the overall loop delay a dummy frame is sent to the receiver and the time is measured between the falling edge of the VALID signal and the rising edge of the READY signal (if the READY signal does not rise after a certain while, the receiver might be defect, not correctly connected or not powered).

The signaling of the READY signal on the receiver side takes place in the clock domain of the transmit clock. The reset value of the TCR.MDP is 0 in the transmitter and in the receiver and as a result, the READY signal will be set high immediately after receiving the parity bit by the receiver. In this case (MDP = 0), the signaling of READY is identical

for a parity error and for a correct frame. Therefore, the number of transmit clock cycles between VALID becoming 0 until READY becoming 1 represents the overall loop delay. It is indicated by the counter bit field TSTATR.RDC (ready delay count). This bit field is a counter starting from 0 each time VALID becomes low (1 to 0 transition) and that stops when READY becomes 1 (0 to 1 transition). It holds the value until the next 1 to 0 transition of VALID is reached. This value can be read out to determine an appropriate MDP value.

The desired value for MDP must be transferred to the receiver. Therefore, the command pipe 1 can be used. A frame of command pipe 1 transfers the value and the receiver stores it automatically as its delay for parity error (RCR.DPE). In order to verify the correct setting of DPE of the receiver, dummy frames are transferred with and without parity error. The parity generation of the receiver always starts with 0 and this value is toggled each time a 1 is received. This result is compared to the received parity bit (the received parity bit does not modify the receivers parity check bit). The parity start bit of the transmitter can be programmed. Assuming a correct transfer, the start value of 1 in the transmitter, will lead to a parity error detection on receiver side. This event is signaled by the receiver by setting READY to 1 after the time indicated by RCR.DPE (in the receiver) has elapsed. By reading the value of TSTATR.RDC, the transmitter software can detect if the receiver's DPE has been set up correctly. In case of an error, the transfer of DPE by the command pipe 1 must be started again, until the results are correct.

All these setup actions should take place while the MLI move engine of the receiver is switched off (automatic mode disabled, receiver in listen mode). The complete setup can be done under the control of the transmitter. A special software on receiver side is not required. If the required values are known on both sides (transmitting and receiving controller), the normal transfers can start.

Note: A dummy frame can be any frame that does not lead to a hardware action in the receiver. In listen mode, the CPU on receiver side should ignore the reception of dummy frames. In order to start normal operation, the transmitter can switch on the automatic mode of the receiver's move engine or send a command via pipe 3 that is then taken into account by the receiver's CPU.

27.1.6 MLI Kernel and MLI Interface Logical Connection

The MLI transmitter will have the possibility of getting 32 address bits, 32 bits of data, a selection for read or write operation, and a selection of the transmission window. The MLI receiver may provide the MLI interface 32 bits of address, 32 bits of data and the type of the transaction (read or write). In addition there will be a signal to request the DMA from the MLI receiver.

Figure 27-15 shows a graphical representation of this interface.

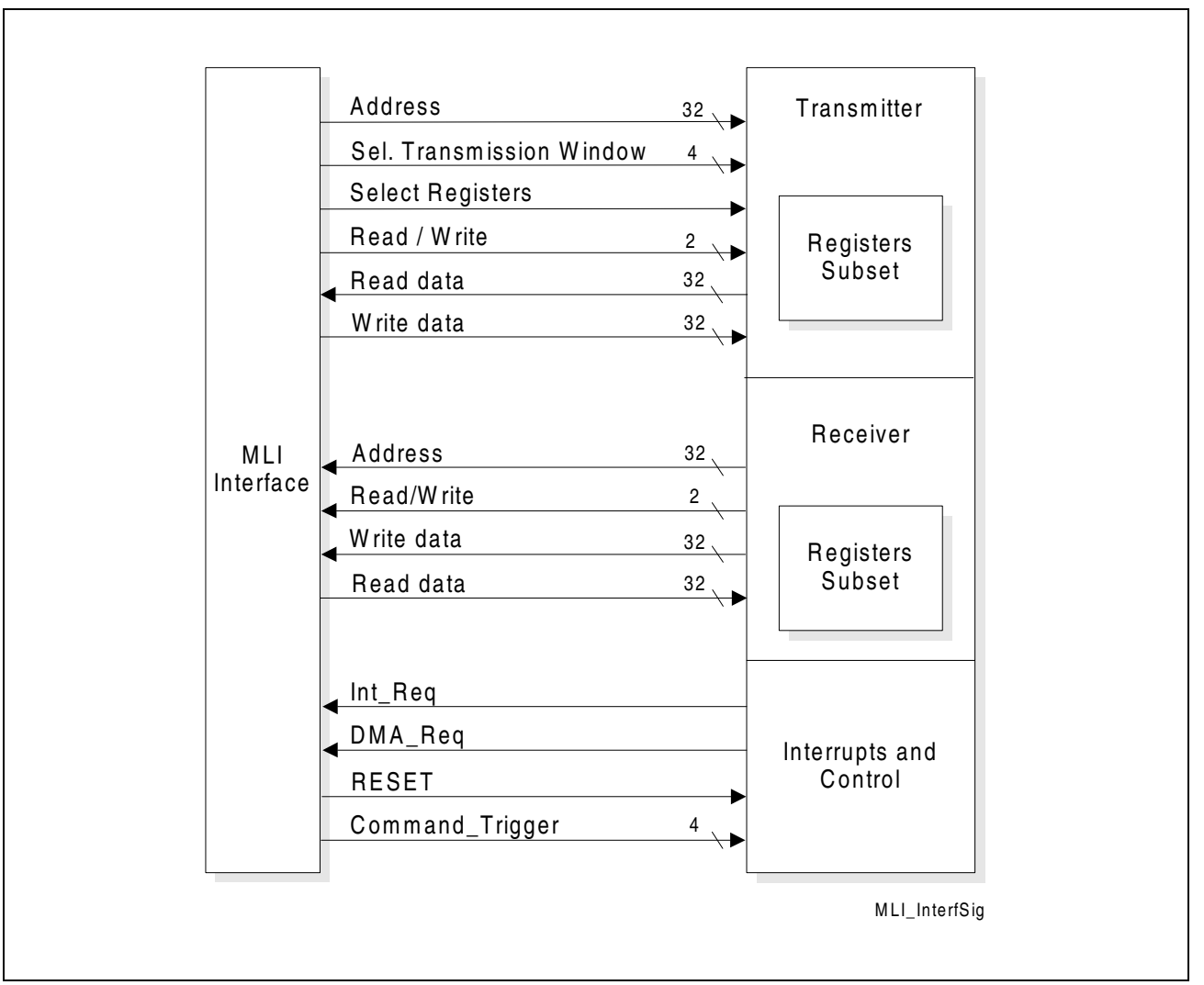


Figure 27-15 Signals Between MLI Interface and MLI Kernel

27.1.7 MLI Transmitter

27.1.7.1 MLI Transmitter Reset

After the hardware reset the MLI transmitter will be in transmitter off mode as it will be explained in **Section 27.1.7.2**. In this state the transmitter will not raise its VALID signal and therefore no transaction will be performed.

27.1.7.2 MLI Transmitter Operation Modes

By programming the MLI transmitter control register it is possible to set its operation mode. In **Table 27-1** are shown all the possible modes of the MLI transmitter, depending on the values of the mode of operation parameter (TCR.MOD). The characteristics of the different modes are explained below.

Table 27-1 MLI Transmitter Operation Modes

TCR.MOD	Mode of Operation
0	MLI transmitter off
1	MLI transmitter on

Transmitter Off

This is the mode in which the MLI transmitter is after the hardware reset. It has the following characteristics:

- The VALID signal is not activated. As a consequence, no transfer is supported.

Transmitter On

Its characteristics are the following ones:

- The whole functionality of the MLI transmitter is available.

Note: The next section will explain the whole MLI transmitter functionality and its interfaces arbitration for the transmitter on operation mode. For each case, it must be taken into account the operation mode in which the MLI transmitter is programmed and consider the limitations introduced by each of the explained modes.

27.1.7.3 Internal Architecture and Interface Signals

Figure 27-16 illustrates the internal architecture of the MLI transmitter:

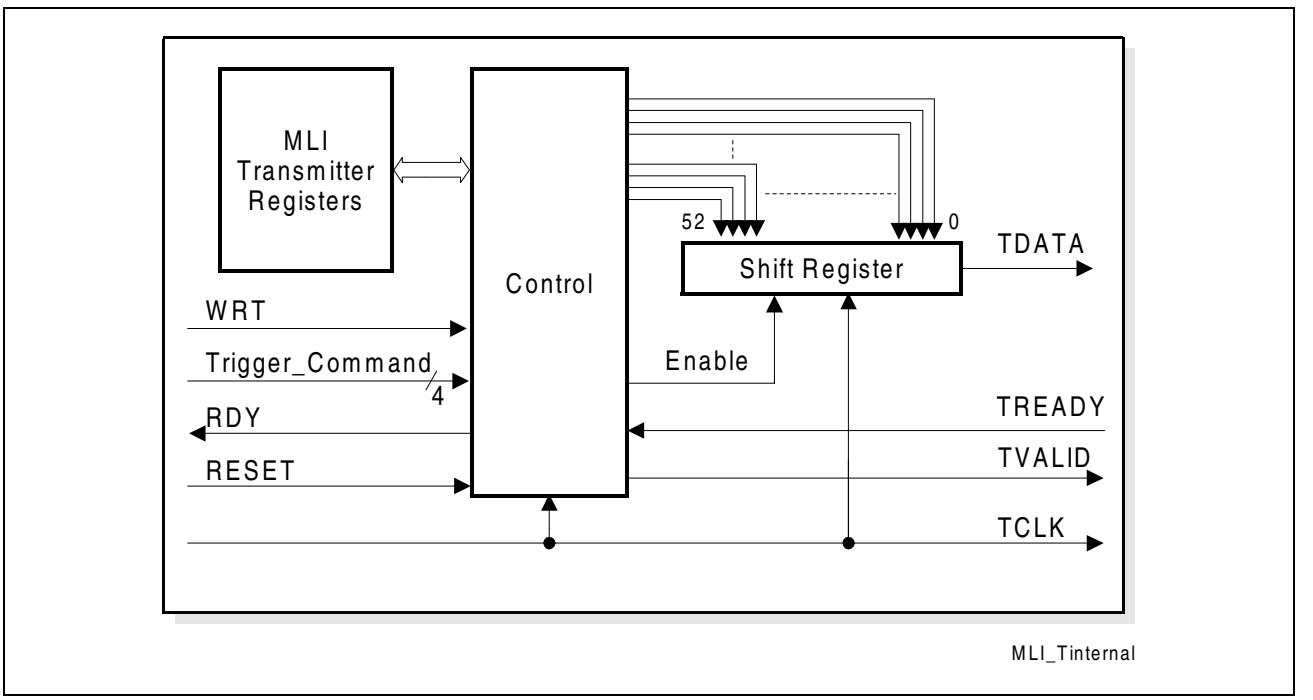


Figure 27-16 MLI Transmitter Internal Architecture

Note: The letter "x" indicates the pipe number, from 0 to 3.

The signal RDY is used to notify the controller that the MLI transmitter is prepared to receive new information in the data and address offset registers (TPxDATAR and TPxAOFR). The 'active' WRT signal indicates that a write access is performed in the MLI registers. The enable signal will control the shift register.

The four trigger command lines will be programmed via hardware the MLI transmitter to send up to four different commands.

The signals TDATA, TREADY, TVALID and TCLK will follow the scheme explained in **Section 27.1.4**.

The MLI transmitter will operate as an information moving agent between two controllers. It will receive through its controller interface side, address offset and data, only address offset or only data. These values will be written in TPxDATAR and TPxAOFR registers by software via the DMA switch, or by the DMA itself.

The MLI transmitter will keep track of the next information:

- Current address offset in the pipe (TPxAOFR, where x denotes the pipe number)
- Width of the current address offset in each of the pipe's address offset registers (TPxSTATR.BS, where x denotes the pipe number)
- Current data in the pipe (TPxDATAR, where x denotes the pipe number)

- Width of the current data received in each of the pipe's data registers (TPxSTATR.DW, where x denotes the pipe number)

A complete list of the MLI transmitter registers may be found in **Section 27.2.1**.

27.1.7.4 Transmission Format

The MLI transmitter first transmits four bits as header (denoted as H) that contains information about the mode of transfer and which is the current pipe in use.

The first two bits in the header will determine what type of transmission it is and it will be referenced as frame code (denoted as FC in the whole chapter). These two bits together with the number of bits of the frame will determine the type of transmission. **Table 27-2** shows its encoding.

Table 27-2 First Two Bits Header Encoding, Frame Code

Header Bits 0 and 1 (FC)	Type of Frame
00_B	Copy base address frame
01_B	Write in offset and data frame or Discrete read frame
10_B	Command frame or Answer frame
11_B	Optimized write frame or Optimized read frame

The second two bits field, indicate which is the current pipe in use. These two bits will be referenced as pipe number (PN) in the whole chapter. **Table 27-3** shows its encoding.

Table 27-3 Second Two Bits Header Encoding, Pipe Number

Header Bits 2 and 3, (PN)	Pipe in Use
00_B	Pipe 0
01_B	Pipe 1
10_B	Pipe 2
11_B	Pipe 3

After this header the MLI transmitter sends the information as it will be explained in **Section 27.1.7.5**. In every case a parity bit will be transmitted in the last position of the frame (p). **Figure 27-17** illustrates parts of the transmission frame.

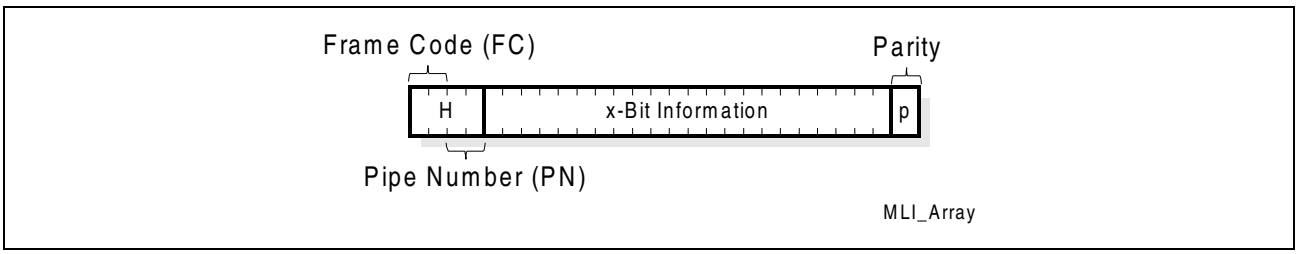


Figure 27-17 Parts of the Transmission Frame

27.1.7.5 Transmission Modes

In this section are explained the different modes in transmission, indicating for each one:

- the frame composition
- where are the values of each field of the frame taken from
- the number of bits that each frame has

Section 27.1.7.6 will explain how the MLI transmitter chooses between these different options of transmission.

Copy Base Address Frame

Its frame code is 00_B. This mode will allow the MLI receiver in the other controller to know the base addresses and the buffer size of each transfer window. **Figure 27-18** illustrates the frame sent by the MLI transmitter.

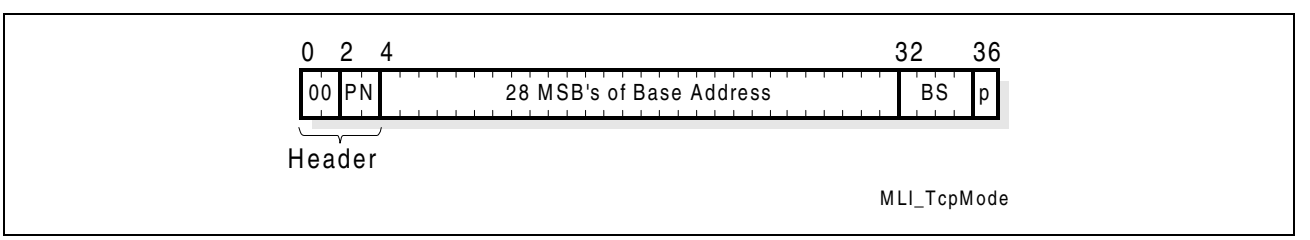


Figure 27-18 Copy Base Address Frame

The frame contains the 28 more significant bits of the base address and the buffer size of this transfer window (the 4 bits denoted as BS). **Table 27-4** illustrates where each of the fields of the frame are taken from.

Table 27-4 Storage of the Values Used in the Frame

Field	Value Taken From
PN	TRSTATR.PN
Base Address	28 MSBs of TCBAR
Buffer Size	TPxSTATR.BS

Note: x indicates the pipe number, x = 0, 1, 2, 3.

The number of bits transmitted in this mode is shown in **Table 27-5**.

Table 27-5 Number of Bits in Copy Base Address Frame

Header	Base Address	Parity	Total
4 bits	32 bits	1 bit	37 bits

Command Frame

Its frame code is 10_B. The MLI transmitter sends the following frame:

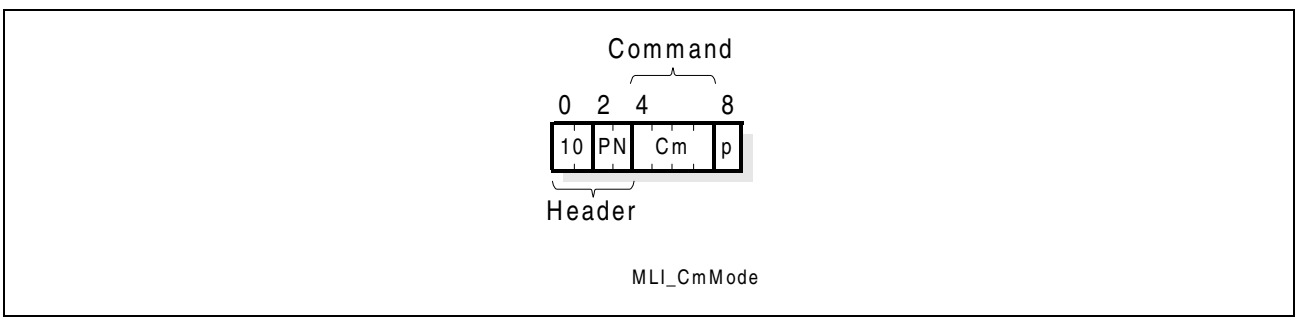


Figure 27-19 Command Frame

The bits denoted as “Cm” in **Figure 27-19** represent the command. Its value is obtained from the command bit field contained in the MLI transmitter command register (TCMDR.CMDPx, where x denotes the pipe number). The meaning of the command depends on the pipe used.

The command is a subset of four bits, which encoding is shown in **Table 27-22**. The actions indicated are performed by the MLI receiver of the second controller. In this table, DPE stands for delay for parity error (stored in RCR.DPE). It indicates the number of TCLK clock periods that the MLI receiver must wait without raising again the READY signal to inform that a parity error was detected.

Table 27-6 illustrates where each of the fields of the frame are taken from.

Table 27-6 Storage of the Values Used in the Frame

Field	Value Taken From
PN	The correspondent to the accessed part of the command register
CM	TCMDR.CMDPx

Note: x indicates the pipe number, x = 0, 1, 2, 3.

The number of bits transmitted is described in **Table 27-7**.

Table 27-7 Bits Transmitted in Command Frame

Header	Command	Parity	Total
4 bits	4 bits	1 bit	9 bits

Write Access To Transfer Window

- Write in Offset and Data Frame: the address offset cannot be predicted

Its frame code is 01_B. The MLI transmitter sends the offset and data frame. **Figure 27-20** illustrates this transfer. The number of bits of the address offset (m) and the bits of data (n) are known and specified in the transmitter status register of the current pipe.

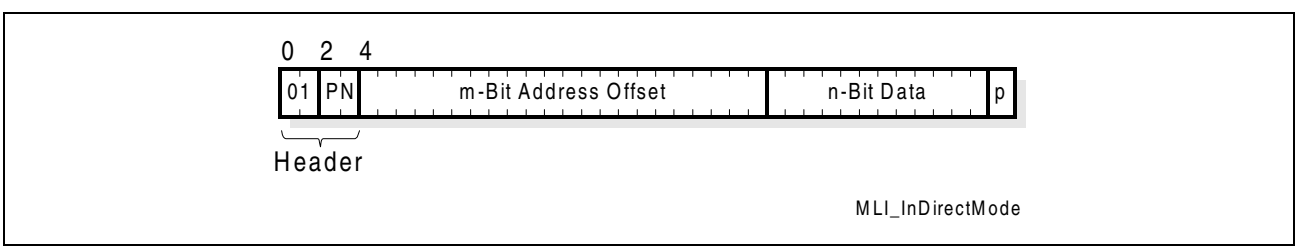


Figure 27-20 Write Access in Offset and Data Frame

Table 27-8 illustrates where each of the fields of the frame are taken from.

Table 27-8 Storage of the Values Used in the Frame

Field	Value Taken From
PN	The correspondent to the accessed registers
Address Offset	TPxAOFR
Data	TPxDATAR

Note: x indicates the pipe number, x = 0, 1, 2, 3.

The number of bits transmitted is shown in **Table 27-9**.

Table 27-9 Number of Bits In Offset And Data Frame

Data Width	Header	Offset	Data	Parity	Total
8 bits	4 bits	m bits	8 bits	1 bit	13+m bits
16 bits	4 bits	m bits	16 bits	1 bit	21+m bits
32 bits	4 bits	m bits	32 bits	1 bit	37+m bits

- Optimized Write Frame: the new address offset can be predicted.

Its frame code is 11_B. **Figure 27-21** shows this transmission mode.

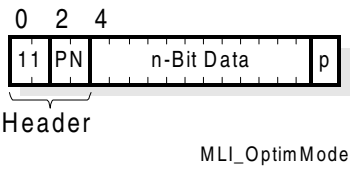


Figure 27-21 Optimized Write Frame

The offset will be deducted in the MLI receiver of the second controller depending on the address prediction factor associated to the current pipe, TPxSTATR.AP in the transmitter side and RPxSTATR.AP in the receiver side.

Table 27-10 illustrates where each of the fields of the frame are taken from.

Table 27-10 Storage of the Values Used in the Frame

Field	Value Taken From
PN	The correspondent to the accessed registers
Data	TPxDATAR

Note: x indicates the pipe number, x = 0, 1, 2, 3.

The number of bits transmitted is shown in **Table 27-11**.

Table 27-11 Number of Bits in Optimized Write Frame

Data Width	Header	Data	Parity	Total
8 bits	4 bits	8 bits	1 bit	13 bits
16 bits	4 bits	16 bits	1 bit	21 bits
32 bits	4 bits	32 bits	1 bit	37 bits

Read Access To Transfer Window

- Discrete Read Frame: the new offset address cannot be predicted.

Its frame code is 01_B. The MLI transmitter sends the offset it wants to read from, indicating the width of the data. The frame in this case is shown in **Figure 27-22**.

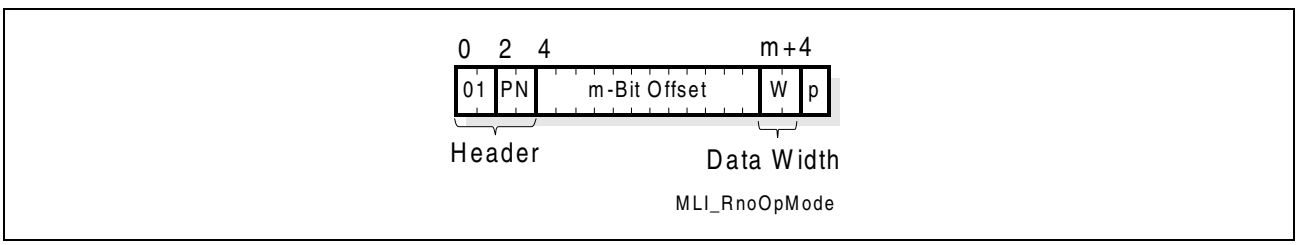


Figure 27-22 Discrete Read Frame

The field referenced as “W” (data width) in **Figure 27-22** indicates the size of data that the first controller wants to read.

Table 27-12 shows the encoding of this field.

Table 27-12 Data Width Encoding

W	Data Width
00_B	8 bits
01_B	16 bits
10_B	32 bits
11_B	Reserved

Table 27-13 illustrates where each of the fields of the frame are taken from.

Table 27-13 Storage of the Values Used in the Frame

Field	Value Taken From
PN	The correspondent to the accessed registers
W	TPxSTATR.DW
Address Offset	TPxAOFR

Note: x indicates the pipe number, x = 0, 1, 2, 3.

The number of bits transmitted is shown in **Table 27-14**.

Table 27-14 Number of Bits In Discrete Read Frame

Header	Data Width	Offset	Parity	Total
4 bits	2 bits	m bits	1 bit	7+m bits

Note: This case is perfectly distinguishable from the write in offset and data frame because the value of the buffer size (TPxSTATR.BS = m, where x indicates the current pipe) is fixed all the time for each pipe.

- Optimized Read Frame: the new address offset can be predicted.

Its frame code is 11_B. The frame in this case is shown in **Figure 27-23**.

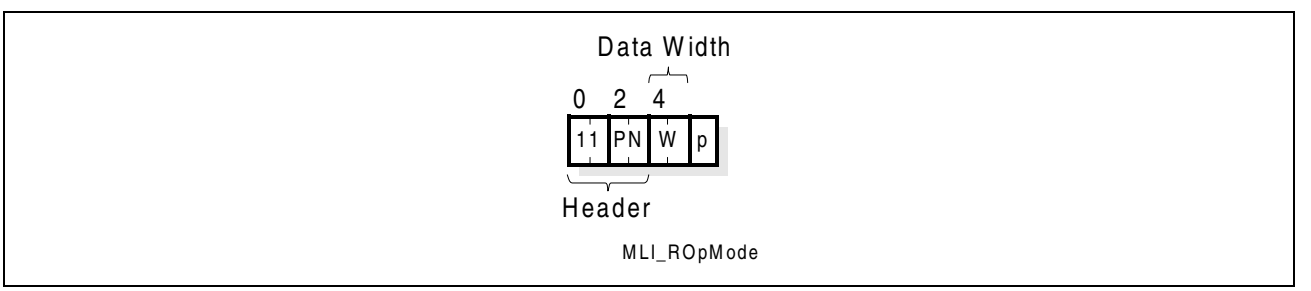


Figure 27-23 Optimized Read Frame

The bit field data width (W) has the same meaning as explained in **Table 27-12**.

Table 27-15 illustrates where each of the fields of the frame are taken from.

Table 27-15 Storage of the Values Used in the Frame

Field	Value Taken From
PN	The correspondent to the accessed registers
W	TPxSTATR.DW

Note: x indicates the pipe number, x = 0, 1, 2, 3.

The number of bits transmitted is shown in **Table 27-16**.

Table 27-16 Number of Bits In Optimized Read Frame

Header	Data Width	Parity	Total
4 bits	2 bits	1 bit	7 bits

Answer Frame

Its frame code is 10_B. **Figure 27-24** illustrates the frame sent by the MLI transmitter.

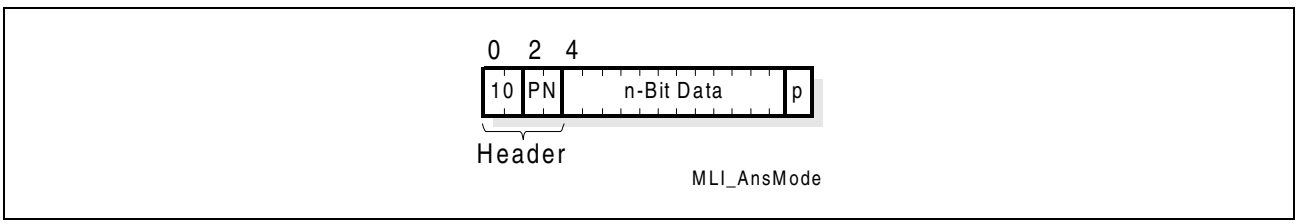


Figure 27-24 Answer Frame

The frame contains a data bit field that is the answer to a read operation made before. **Table 27-17** illustrates where each of the fields of the frame are taken from.

Table 27-17 Storage of the Values Used in the Frame

Field	Value Taken From
PN	TSTATR.APN
Data	TDRAR

The answer frame may be sent through any pipe because in any moment there will be only one read operation in course.

The number of bits transmitted is shown in **Table 27-18**.

Table 27-18 Number of Bits In Answer Frame

Data	Header	Parity	Total
8 bits	4 bits	1 bit	13 bits
16 bits	4 bits	1 bit	21 bits
32 bits	4 bits	1 bit	37 bits

27.1.7.6 Transfer Mode Selection

Each time a write access to the MLI registers is made, a related flag is set in the transmitter status register (TRSTATR), indicating that the data stored in them has not been sent (transfer pending) and no more writing accesses may be made on these registers. These flags are reset again when the registers may be written again.

Table 27-19 shows the flags that are set when accessing the different registers and when they are reset again.

Table 27-19 Valid Flags

Flag	Set When Access to	Reset When
TRSTATR.DVx	TPxAOFR, TPxDATAR for read or write operation	The write or read frame has been sent correctly through the correspondent pipe
TRSTATR.RPx	TPxAOFR, TPxDATAR for read operation	The answer frame has been received correctly through the correspondent pipe
TRSTATR.CIVx	The MLI transmitter detects a rising edge in the trigger_commandx line	The correspondent command frame has been sent correctly through the pipe 0
TRSTATR.CVx	TCMDR or CVx is set via software	The command frame has been sent correctly through the correspondent pipe
TRSTATR.BAV	TCBAR	The copy base address frame has been sent correctly through the correspondent pipe
TRSTATR.AV	TDRAR	The answer frame has been sent correctly through the correspondent pipe

Note: x indicates the pipe number, x = 0, 1, 2, 3.

When a rising edge is detected in the trigger_commandx line, the MLI transmitter will set the correspondent CIVx bit, and it will send a command frame through pipe 0 as explained in **Table 27-20**.

Table 27-20 Hardware Triggered Command

Rising Edge	Command to Send Through Pipe 0
Trigger_command0	0001_B
Trigger_command1	0010_B
Trigger_command2	0011_B
Trigger_command3	0100_B

When more than one transfer is pending, the criteria to choose between the different frames to send will be based on a priority scheme as follows:

- Read answer frame, command frame triggered by hardware, command frame triggered by software, read frame, write frame and copy base address frame.
- If two or more pipes operations are pending with the same priority, then the one with the highest priority will be chosen. The pipe number 0 has the highest priority, and then pipe 1, 2 and 3 by this order.

The following paragraphs illustrate the different actuations taken depending on the type of frame chosen to be sent:

Copy Base Address Frame

The TRSTATR.BAV flag is set to one, and there is no other type of frame pending in this pipe. This transfer means that the controller wants to initialize the corresponding pipe base address of the MLI receiver in the other controller and its correspondent buffer size. The MLI transmitter will send the copy of base address frame with the base address stored in the TCBAR register and the buffer size stored in the correspondent TPxBAR register.

Write and Read Frame

- If only the TRSTATR.DVx flag is set, it indicates a write operation.
- If TRSTATR.DVx and TRSTATR.RPx flags are set, they indicate a read operation.

Depending on the operation code this will mean that the first controller wants to write the data (stored in TPxDATAR) in the relative address indicated by the address offset (that will be stored in TPxAOFR) of the transfer window defined by the selected pipe or it wants to read data from that position.

If the address prediction method is allowed (TCR.NO = 0, where x indicates the pipe) then the MLI transmitter will compare the new address offset written in the bus with the old one (TPxAOFR in the moment of accessing). The difference between these two addresses is stored in the correspondent pipe status register (TPxSTATR.AP) if it is not greater than 10 bits (in two’s complement). If the difference is the same in two consecutive transfers then the MLI transmitter will set the optimized flag in the status register of the correspondent pipe (TPxSTATR.OP). This will indicate that the optimized

mode will be used to send this frame when it was chosen to be sent. The optimized mode will be used for the rest of the frames, as far as TPxSTATR.OP = 1, until the difference was not the same as the one stored in the status register, (then the MLI resets TPxSTATR.OP again).

When the conditions explained above are not met, then the transfer is performed using a normal frame for writing or reading (discrete read frame or write in address offset and data frame).

When the necessary comparisons to infer the address prediction factor, the new address offset is finally stored in the address offset register of the pipe (TPxAOFR).

Next pseudocode illustrates the process:

```
if (TPxSTATR.NO = 0) then                -- Optimized mode possible
delta = new_address_offset - TPxAOFR
 if (delta = TPxSTATR.AP) then
  if (Coincidence = TRUE) then
   TPxSTAT.OP = 1
  else
   Coincidence = TRUE
   TPxSTAT.OP = 0
  end if
 else if (if difference delta is not bigger than 9 bits) then
  TPxSTATR.AP = delta
  Coincidence = FALSE
  TPxSTAT.OP = 0
 else
  Coincidence = FALSE
  TPxSTAT.OP = 0
 end if
else                                     -- Optimized mode disabled
 TPxSTAT.OP = 0
end if
```

Note: TPxAOFR contains the last address offset used in the pipe. TPxSTATR.AP is the address prediction factor. The boolean variable Coincident, expresses the condition of two consecutive coincidences in the address offsets.

When the data of the pipe was chosen to be sent, using the priority method explained before, the MLI will know whether the frame should or not be sent using the optimized mode.

When the write or read frame is correctly received by the other controller, the MLI resets the correspondent TRSTATR.DVx flag. In the case of a read operation, the flag TRSTATR.RPx will be reset by hardware when the answer frame was received. Until this read pending flag is reset again, the registers TPxDATAR and TPxAOFR of the pipe may be written again.

Note: Users who prefer to use the optimized read/write frames must not use the small and large MLI transfer windows in a single pipe at the same time.

Answer Data Frame

This means that the written data in the TDRAR register is the answer to a read operation, therefore the transmission will be made in answer frame.

When the writing frame is correctly received by the other controller, the MLI resets the TRSTATR.AV flag.

Command Frame

If CIVx is set, the MLI transmitter will send the command frame correspondent to the line through the pipe 0 as explain in **Table 27-20**. When the command frame is correctly received by the other controller, the MLI resets the correspondent TRSTATR.CIVx flag.

If CVx is set, the MLI transmitter will check the value stored in the command bit field correspondent to the pipe of the TCMDR register and it will send it using the command frame. When the command frame is correctly received by the other controller, the MLI resets the correspondent TRSTATR.CVx flag.

27.1.7.7 Parity Generation

The type of parity used is even or odd parity, depending on the programmed value in TCR.TP. The parity bit is calculated by toggling a bit each time that a one is sent in the frame. The starting value for the toggling bit is zero if the parity is even and is one if the parity is odd. Assuming a correct transmission, a starting value of 1 in the toggling bit will lead into a parity error situation as explained in **Section 27.1.5**.

27.1.7.8 Error Detection and Handling

The MLI transmitter will be able to recognize the following error situations in the transmission:

- a non acknowledged transfer
- a parity error

Non Acknowledge Error Detection

This situation is explained in **Figure 27-12** and in **Figure 27-40**. When this error is detected, the MLI transmitter will set the non acknowledge error flag (TSTATR.NAE) and decreases the counter of non acknowledge errors (TCR.MNAE). When this counter reaches the value zero, a time-out interrupt is generated if enabled.

Parity Error Detection

The Parity Error situation is explained in **Figure 27-13** and in **Figure 27-39**. When this error is detected, the MLI transmitter will set the parity error flag (TSTATR.PE) and decreases the counter of parity errors (TCR.MPE). When this counter reaches the value zero, a parity error interrupt is generated if enabled.

Whenever the MLI transmitter sends correctly a new frame, it will produce an interrupt. The interrupt is enabled depending on the type of frame (command frame interrupt or normal frame interrupt). Please refer to **Section 27.1.10**.

27.1.7.9 MLI Transmitter Input/Output Control

Figure 27-25 shows the control structure for the transmitter output signals VALID, CLK and DATA. The VALID signal can be distributed to up to four output lines (TVALIDA to TVALIDD). It is possible to individually enable/disable each line (except DATA) and to select its polarity.

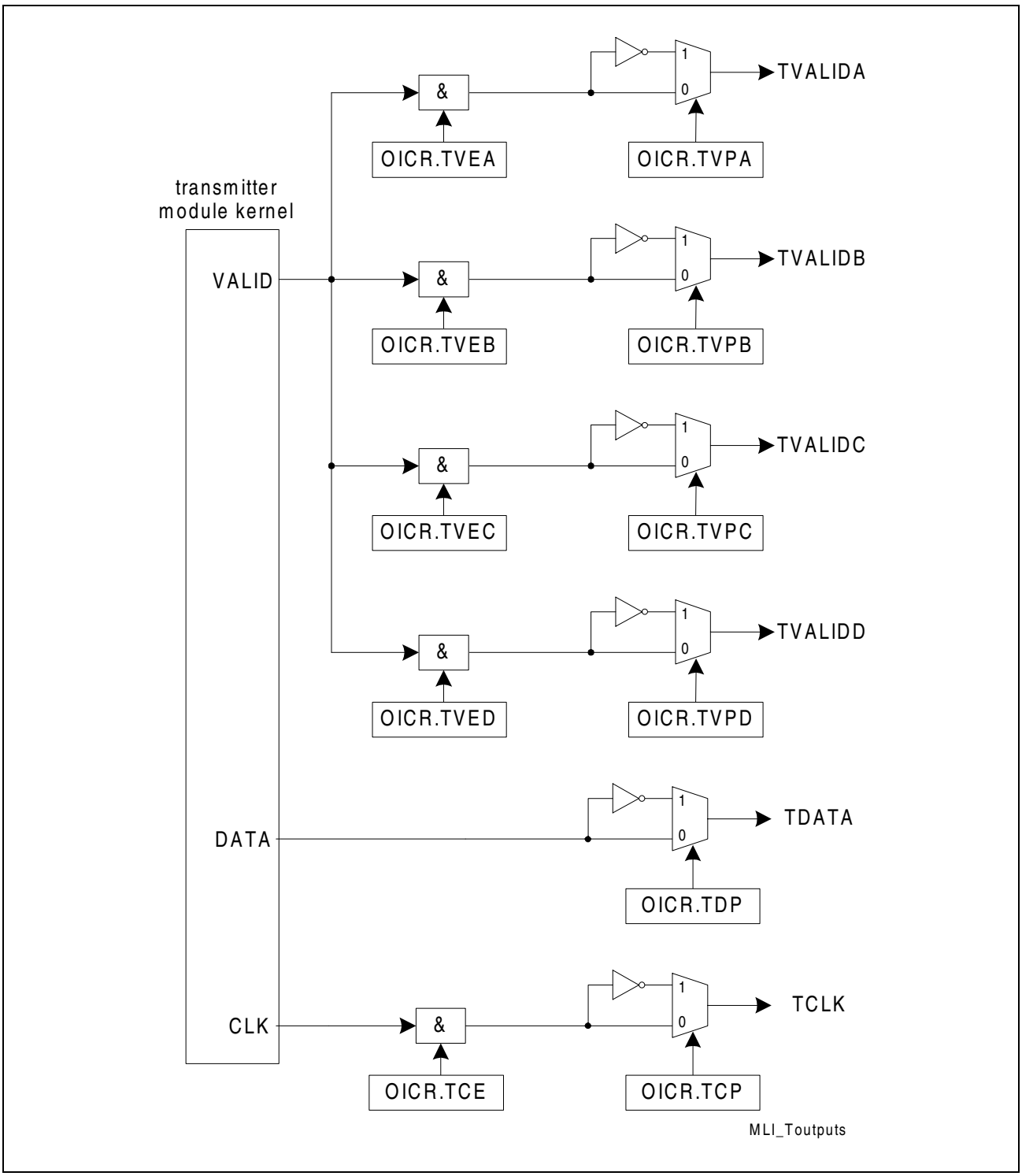


Figure 27-25 Control of the Transmitter Output Signals

The transmitter output shift clock signal CLK can be enabled by the bit OICR.TCE and its output polarity can be selected by OICR.TCP. For the data signal DATA it is possible to select the polarity by programming bit OICR.TDP.

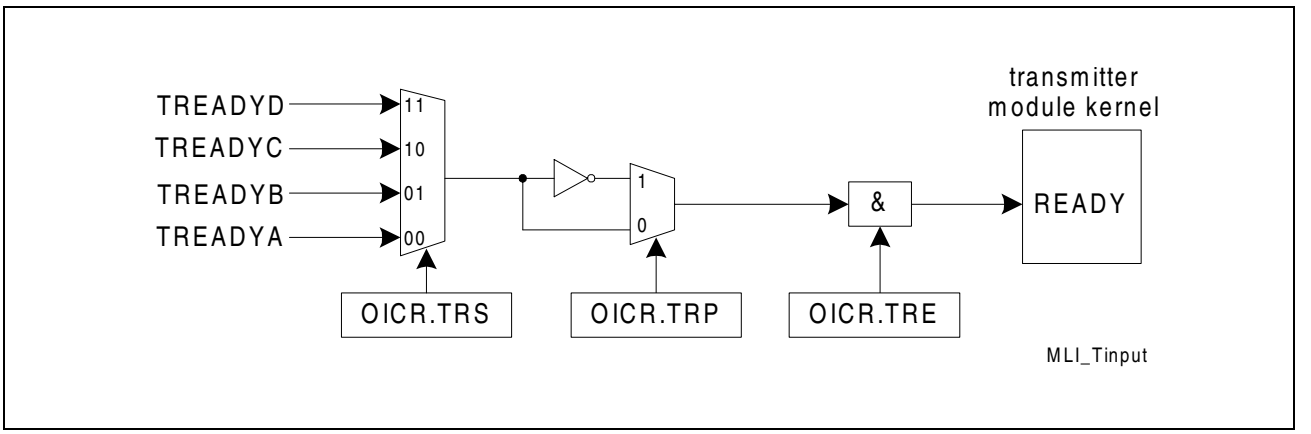


Figure 27-26 Control of the Transmitter Input Signal

The transmitter input signal READY can be selected from four possible input signals (TREADYA to TREADYD). The selected input signal can be enabled by bit OICR.TRE and its polarity can be chosen by OICR.TRP.

Note: The first letter “T” of the signal names indicates that these signals belong to the transceiver part of an MLI module. The last letter “A” to “D” of a signal belonging to a set of lines indicates that the signal can be selected from (input) or can be distributed to (output) up to 4 lines.

27.1.8 MLI Receiver

27.1.8.1 MLI Receiver Reset

After the hardware reset, the MLI receiver will be in receiver off mode as explained in **Section 27.1.8.2**.

27.1.8.2 MLI Receiver Operation Modes

By programming the MLI receiver control register it is possible to set its operation mode. **Table 27-21** shows all the possible modes of the MLI receiver, depending on the values of the mode parameter (RCR.MOD). The characteristics of the different modes are explained below.

Table 27-21 MLI Receiver Operation Modes

RCR.MOD	Mode of Operation
0	MLI move engine off, receiver off
1	MLI move engine in automatic mode

Move Engine Off

- The Ready signal is activated and the acknowledge and parity error conditions are accomplished.
- All the write or read transfers have no active effect on the FPI bus in the MLI receiver side.
- Modification of the control parameters or the base addresses of the transmission pipes are allowed.
- Commands are taken into account. Possibility of interrupts programming.

Move Engine in Automatic Mode

- The Ready signal is activated and the acknowledge and parity error conditions are accomplished.
- All the data transfers are taken into account.
- Modification of the control parameters or the base addresses of the transmission pipes are allowed.

This bit may be modified by the MLI receiver whenever it receives the proper command.

Note: The next section will explain the whole MLI receiver functionality and its interfaces arbitration for the receiver on operation mode. For each case, it must be taken into account the operation mode in which the MLI receiver is programmed and consider the limitations introduced by each of the explained modes.

27.1.8.3 Internal Architecture and Interface Signals

Figure 27-27 shows the internal architecture of the MLI receiver:

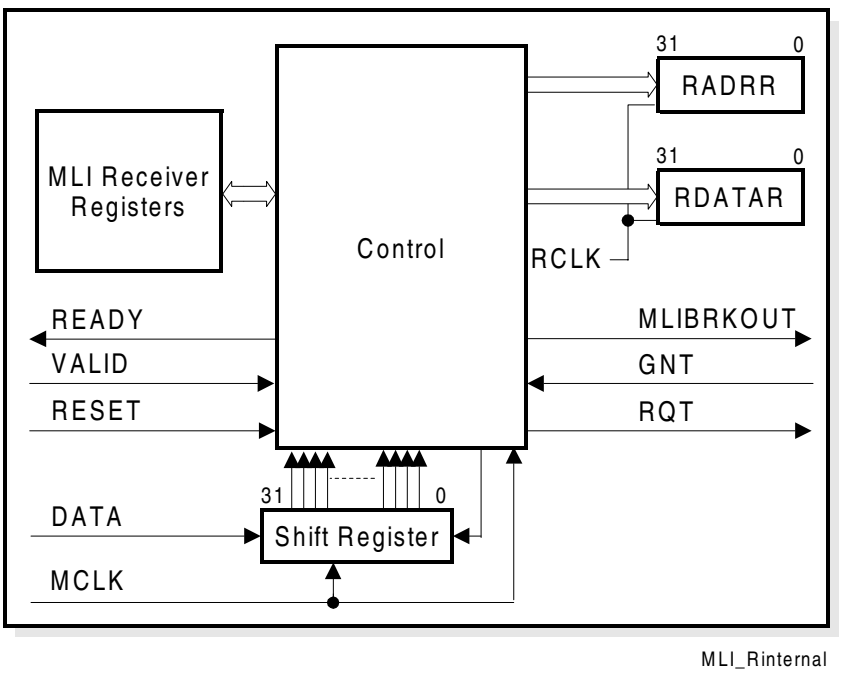


Figure 27-27 MLI Receiver Internal Architecture

Note: The letter "x" indicates the pipe number, from 0 to 3.

The signal RQT is used to notify the DMA that the MLI receiver has new data in its registers RADRR and RDATAR. When the DMA reads the registers RADRR and RDATAR, the MLI interface sets the signal GNT high. From the MLI interface point of view, the signals are explained in **Section 27.1.4**.

Each register RPxBAR (where x indicates the pipe) contains a default value for the base address of each of the four pipes. Each time the MLI receiver obtains a new frame, it will recognize the kind of transmission and the pipe from the header and from the number of received bits. The MLI receiver will keep track of the next parameters:

- Width of the current data received in the pipe (RPxSTATR.DW, where x denotes the pipe number)
- Width of each transfer window (RPxSTATR.BS, where x denotes the pipe number)

A complete list of the MLI receiver registers may be found in **Section 27.2.2**.

27.1.8.4 MLI Receiver Operation

The MLI receiver will obtain the information sent from the MLI transmitter of the other controller. It will operate on the information received in order to extract each of the bit fields. Finally the data and address registers (RDATAR, RADRR) are updated with a new value. Due to the fact that all the data is received synchronized with the clock of the MLI transmitter from the other controller, the MLI receiver must synchronize the address and the data values with its internal clock (RCLK) and the registers RDATAR and RADRR contain their values.

Its operation depends on the type of frame received. A complete reference of different modes is described in **Section 27.1.7.5**.

In order to accomplish the information split, the MLI receiver will take into account the number of bits of the received frame, the transmission mode and the buffer size for the current pipe.

Copy Base Address Frame

Its description and the number of bits of this frame are shown in **Figure 27-18** and in **Table 27-5**, respectively.

After the header, the MLI receiver obtains the 28 MSBs of the current pipe base address, which is stored in the 28 MSBs of its corresponding pipe base address register (RPxBAR). The buffer size is stored in the status register (RPxSTATR.BS) and the type of frame bit field is updated (RCR.TF = 00_B). A normal frame received interrupt is produced if it is enabled by RIER.NFRIE.

Command Frame

Its description and the number of bits of this frame are shown in **Figure 27-19** and in **Table 27-7**, respectively. After the complete reception of this frame, the MLI receiver splits the information in its different bit fields.

Table 27-22 illustrates the different values that may be received in the command frame and the action taken for each one.

Table 27-22 Command Frame Encoding

PN	Cm	Action
00_B	0001_B	Generate interrupt 0, if enabled by RIER.ICE
	0010_B	Generate interrupt 1, if enabled by RIER.ICE
	0011_B	Generate interrupt 2, if enabled by RIER.ICE
	0100_B	Generate interrupt 3, if enabled by RIER.ICE
	Others	No effect

Table 27-22 Command Frame Encoding (cont'd)

PN	Cm	Action
01_B	0000_B	Set RCR.DPE = 0000_B
	0001_B	Set RCR.DPE = 0001_B
	0010_B	Set RCR.DPE = 0010_B
	0011_B	Set RCR.DPE = 0011_B
	…	…
	1111_B	Set RCR.DPE = 1111_B
10_B	0001_B	Set RCR.MOD = 1, enable automatic mode
	0010_B	Set RCR.MOD = 0, disable automatic mode (listen mode)
	0100_B	Reset TRSTATR.RP0
	0101_B	Reset TRSTATR.RP1
	0110_B	Reset TRSTATR.RP2
	0111_B	Reset TRSTATR.RP3
	1111_B	Generate a pulse on line $\overline{\text{MLIBRKOUT}}$ (if enabled by RCR.BEN)
	others	No effect
11_B	Any	Command meaning interpreted by software

When the MLI receiver gets the command 1111_B in pipe 2 (PN = 10_B), then it asserts the signal MLIBRKOUT, which is active in low level if enabled by RCR.BEN bit. **Figure 27-28** illustrates this procedure.

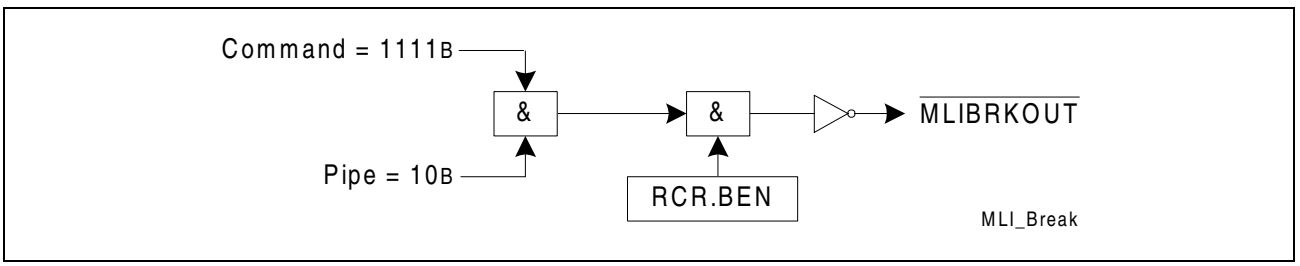


Figure 27-28 Assertion of MLIBRKOUT Signal

Write Access In Offset and Data Frame

Its description and the number of bits of this frame are shown in **Figure 27-20** and in **Table 27-9**, respectively.

After the header, the MLI receiver obtains the m bits corresponding to the offset. The MLI receiver knows how many bits corresponding to the address offset because it is the same as the buffer size of the current pipe (RPxSTATR.BS).

In order to follow the same address prediction method that is carried out by the MLI transmitter, the MLI receiver will compare the address offset of the currently received frame with the address offset previously received. The last address offset is in the receiver base address register of the pipe (RPxSTATR.BS / LSB's of RPxBAR, where x = 0, 1, 2, 3 indicates the pipe). If the difference between both addresses is less than 9 bits, the MLI transmitter will store it in the address prediction factor bit field (RPxSTATR.AP), and this value will be used to obtain the address whenever an optimized frame was received. After this comparison, the newly received address offset is stored in the lowest part of RPxBAR.

Figure 27-29 illustrates this process.

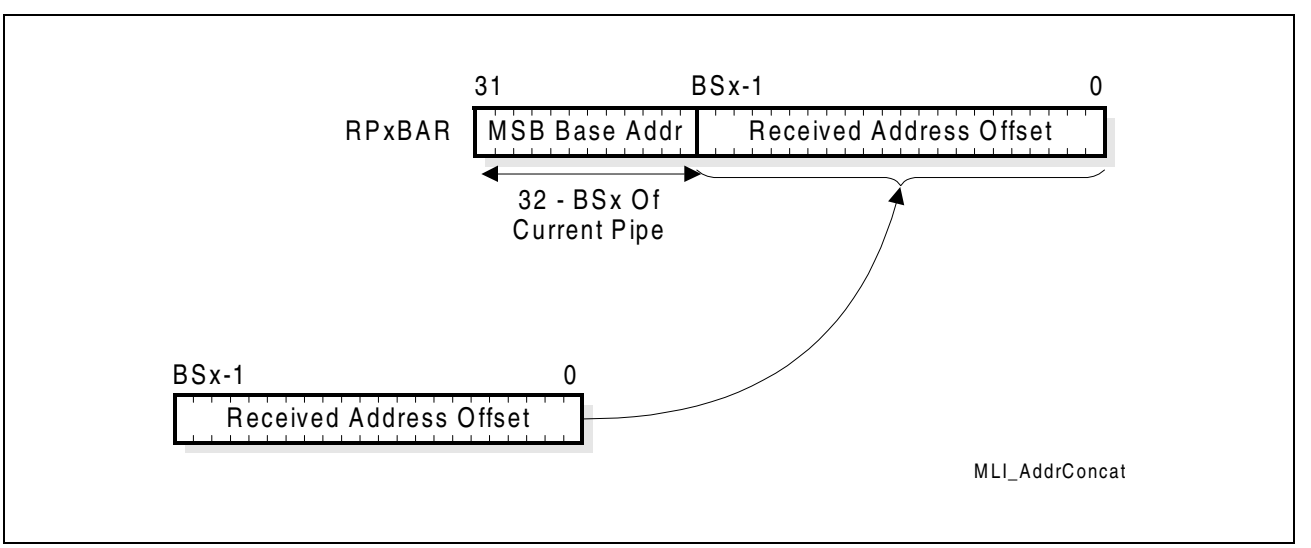


Figure 27-29 Absolute Address Obtaining

Note: BS stands for buffer size (in bits) of the current pipe (x).

After the concatenation, this absolute address will be stored in the MLI receiver address register RADRR. The next subset of bits corresponds to the data and the parity bit. The data will be stored in the RDATAR. The RCR.DW and RCR.TF bit fields are updated, RCR.TF = 10. A normal frame received interrupt is produced if enabled by RIER.NFRIE.

Table 27-23 shows the place of storage of the information obtained from the frame.

Table 27-23 Place of Storage for Address Offset and Data Bits

m - Bit Address Offset	n-Bits Data
BS least significant bits of RPxBAR. After that, copy 32 bits of the register RPxBAR in RADRR.	RDATAR

Optimized Write Frame

Its description and the number of bits of this frame are shown in Figure 27-21 and in Table 27-11, respectively.

After the header, the MLI receiver obtains n bits of data that will be stored in the MLI receiver data register (RDATAR).

The address offset must be calculated by adding to the last address offset used (the content of RPxBAR register) the address prediction factor (RPxSTATR.AP, 10 bits with one sign bit).

Figure 27-30 illustrates how to obtain the address offset.

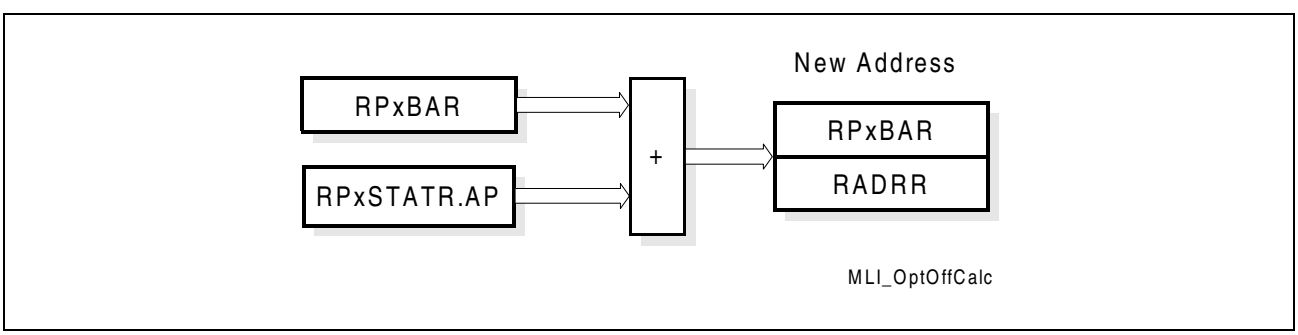


Figure 27-30 Address Offset Prediction for Optimized Write Frame

If the increment or decrement of the address offset results in overflow or underflow, then the wraparound method is used to fit the new address to the transfer window.

The newly obtained 32-bit address is stored in RPxBAR and RADRR registers. The RCR.DW and RCR.TF bit fields are updated, RCR.TF = 10. A normal frame received interrupt is produced if it is enabled by RIER.NFRIE.

Discrete Read Frame

Its description and the number of bits of this frame are shown in **Figure 27-22** and in **Table 27-14**, respectively.

After the header, the MLI receiver obtains two bits, which indicate the width of the data that must be read. The next m bits (as RPxSTATR.BS) represent the offset from which this data will be read from.

In order to follow the same address prediction method that is carried out by the MLI transmitter, the MLI receiver will compare the address offset of the currently received frame with the address offset previously received. This last address offset is in the receiver base address register of the pipe (RPxSTATR.BS / LSB's of RPxBAR, where x = 0, 1, 2, 3 indicates the pipe). If the difference between both addresses is less than 9 bits, the MLI transmitter will store it in the address prediction bit field (RPxSTATR.AP), and this value will be used whenever an optimized frame is received.

The absolute address is calculated by concatenating the obtained address offset with the base address of the current pipe as explained in **Figure 27-29** and then it is stored in the register RADRR. The RCR.DW and RCR.TF bit fields are updated, RCR.TF = 01. A normal frame received interrupt is produced if it is enabled by RIER.NFRIE.

Table 27-24 shows where each of the received bit fields is stored.

Table 27-24 Place of Storage

Data Width	RPxSTATR.BS bits of Address Offset
RPxSTATR.DW	Concatenate with the (32 - RPxSTATR.BS) more significant bits of RPxBAR. Copy this new value of RPxBAR in RADRR.

Optimized Read Frame

Its description and the number of bits of this frame are shown in **Figure 27-23** and in **Table 27-16**, respectively.

After the header, the MLI receiver obtains two bits, which indicate the width of the data that must be read.

The absolute address must be calculated by adding to the last address used (RPxBAR) the address prediction factor (RPxSTATR.AP, 10 bits with one sign bit) as explained in **Figure 27-30**. If the increment or decrement of the address offset results in overflow or underflow, then the wraparound method is used to fit the new address to the transfer window.

The 32-bit resulting absolute address will be stored in the RADRR and RPxBAR registers. The RCR.DW and RCR.TF bit fields are updated, RCR.TF = 10. A normal frame received interrupt is produced if it is enabled by RIER.NFRIE.

Answer Frame

Its description and the number of bits of this frame are shown in **Figure 27-24** and in **Table 27-18**, respectively. If an answer frame is received and the next condition is not met:

- TRSTATR.RPx = 1 and TRSTATR.DVx = 0, where x is the pipe from which the frame is received,

then the frame is discarded and a discarded read answer interrupt is produced if it is enabled by RIER.DRAIE.

After the header, the MLI receiver obtains n bits of data that will be stored in the RDATAR register. The data is the answer to a read operation started by the MLI transmitter of the same MLI. The RCR.DW and RCR.TF bit fields are updated, RCR.TF = 11.

The MLI receiver will then produce a normal frame received interrupt controlled by RIER.NFRIE.

27.1.8.5 Access Protection

An access protection is implemented in the MLI receiver. It prevents undesirable read or write access to parts of the memory map in the receiving controller. The register AER will enable the read and write rights to parts of the memory map, and the register ARR will define the range of memory for parts of the memory.

27.1.8.6 Error Handling

The parity bit is checked using the same method as explained in **Section 27.1.7.7**. If the MLI receiver detects a parity error in the transmission it will raise the READY signal following the next conditions:

- If the MLI receiver is prepared to receive a new frame before the delay that indicates the parity error programmable value (RCR.DPE) clock periods, then it will wait until RCR.DPE clock periods had passed. This DPE value will be received with a command frame from the MLI transmitter in the other controller.
- If the MLI is not prepared before RCR.DPE clock periods had passed, then it will raise the READY signal as soon as it is prepared to receive a new frame.

This will inform the transmitter in the other controller that a parity error is detected. If the parity error is signaled, the MLI receiver sets the parity error flag (RCR.PE) and decreases the counter of parity errors (RCR.MPE). When this counter reaches the value zero, a parity error interrupt is generated if it is enabled.

Figure 27-31 illustrates how the MLI receiver informs the MLI transmitter a parity error.

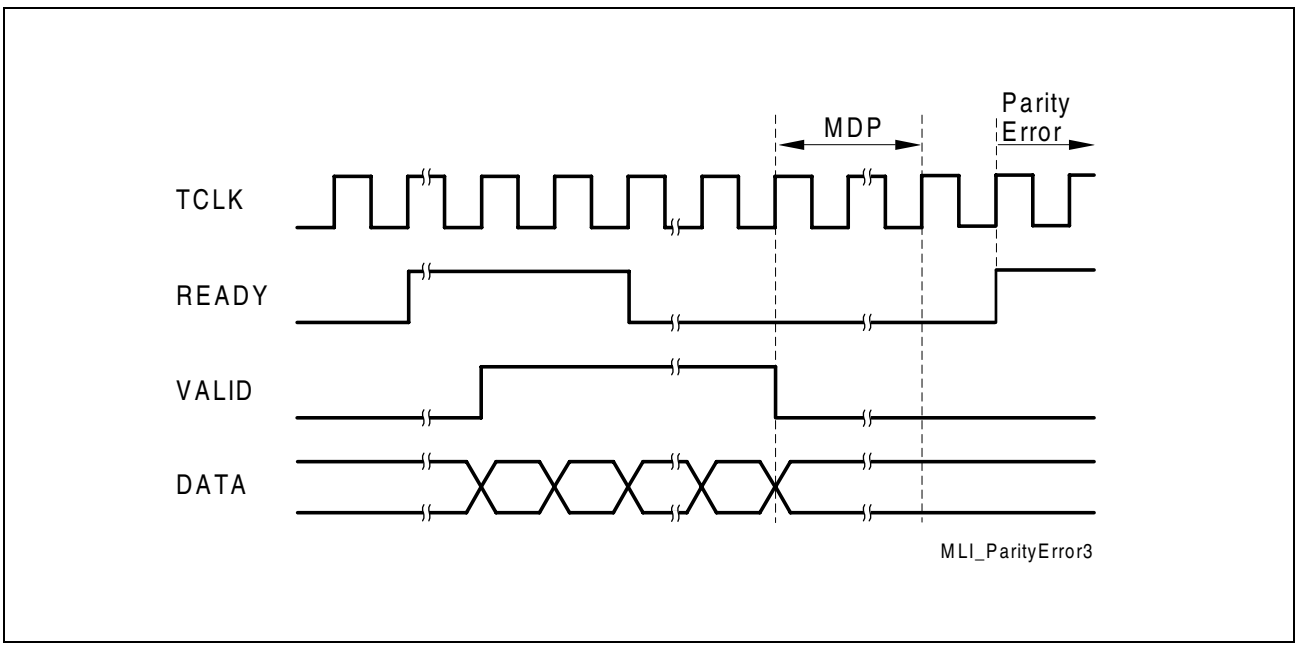


Figure 27-31 Parity Error

Note: The signals are seen from the MLI transmitter side. When VALID is not asserted the DATA line will have only a value (one or zero) depending on the programmed value in TCR.DNT and its chosen polarity.

27.1.8.7 MLI Receiver Input/Output Control

Figure 27-32 shows the control structure for the receiver input signals VALID, CLK and DATA. Each input signal to the receiver module kernel can be selected from up to four input lines. It is possible to individually enable/disable each line (except DATA) and to select its polarity.

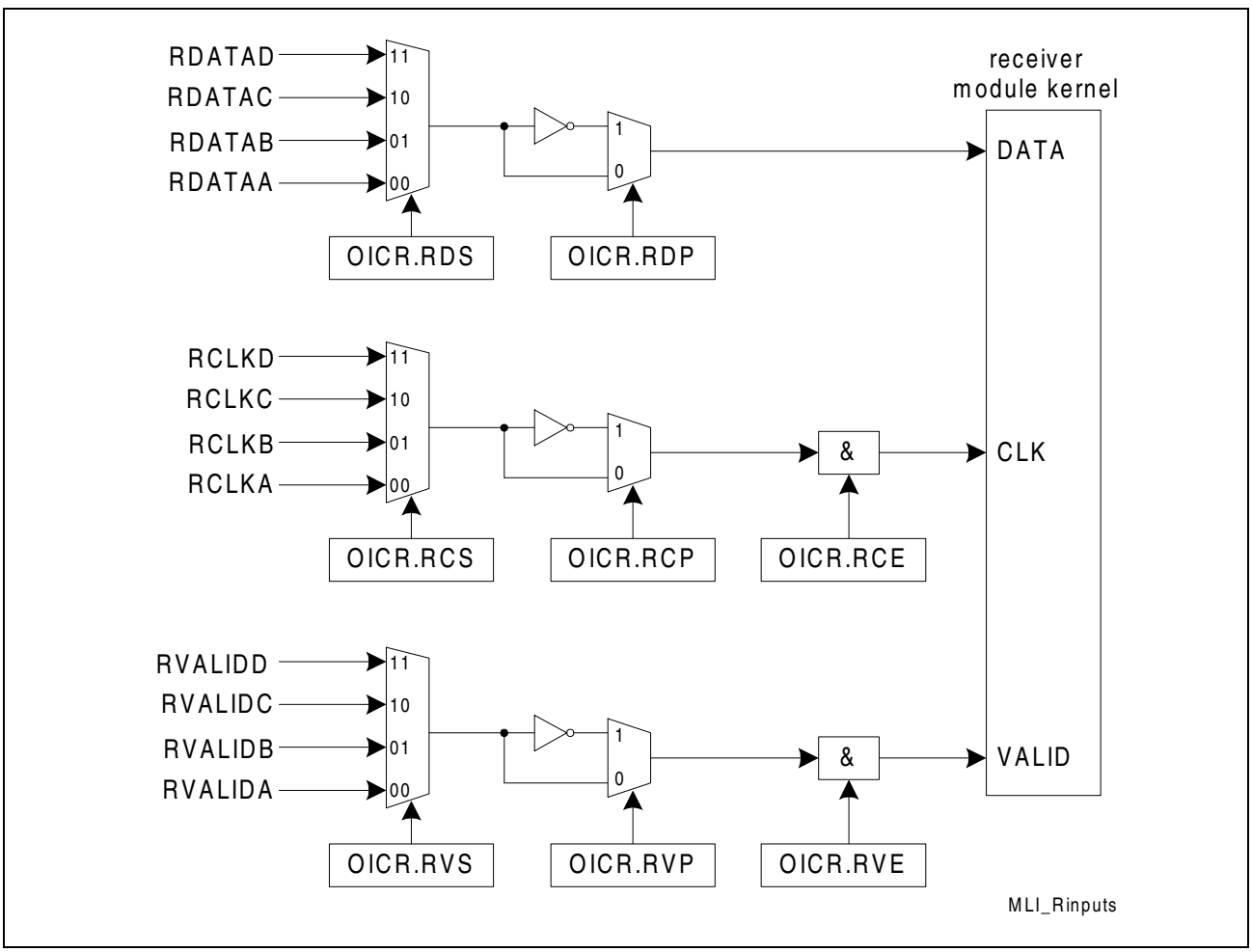


Figure 27-32 Control of Receiver Input Signals

Note: The first letter "R" of the signal names indicates that these signals belong to the receiver part of an MLI module. The last letter "A" to "D" of a signal belonging to a set of lines indicates that the signal can be selected from (input) or can be distributed to (output) up to 4 lines.

The receiver output signal READY can be distributed to up to four output lines (RREADYA to READYD). It is possible to individually enable/disable each line and to select its polarity. See **Figure 27-33**.

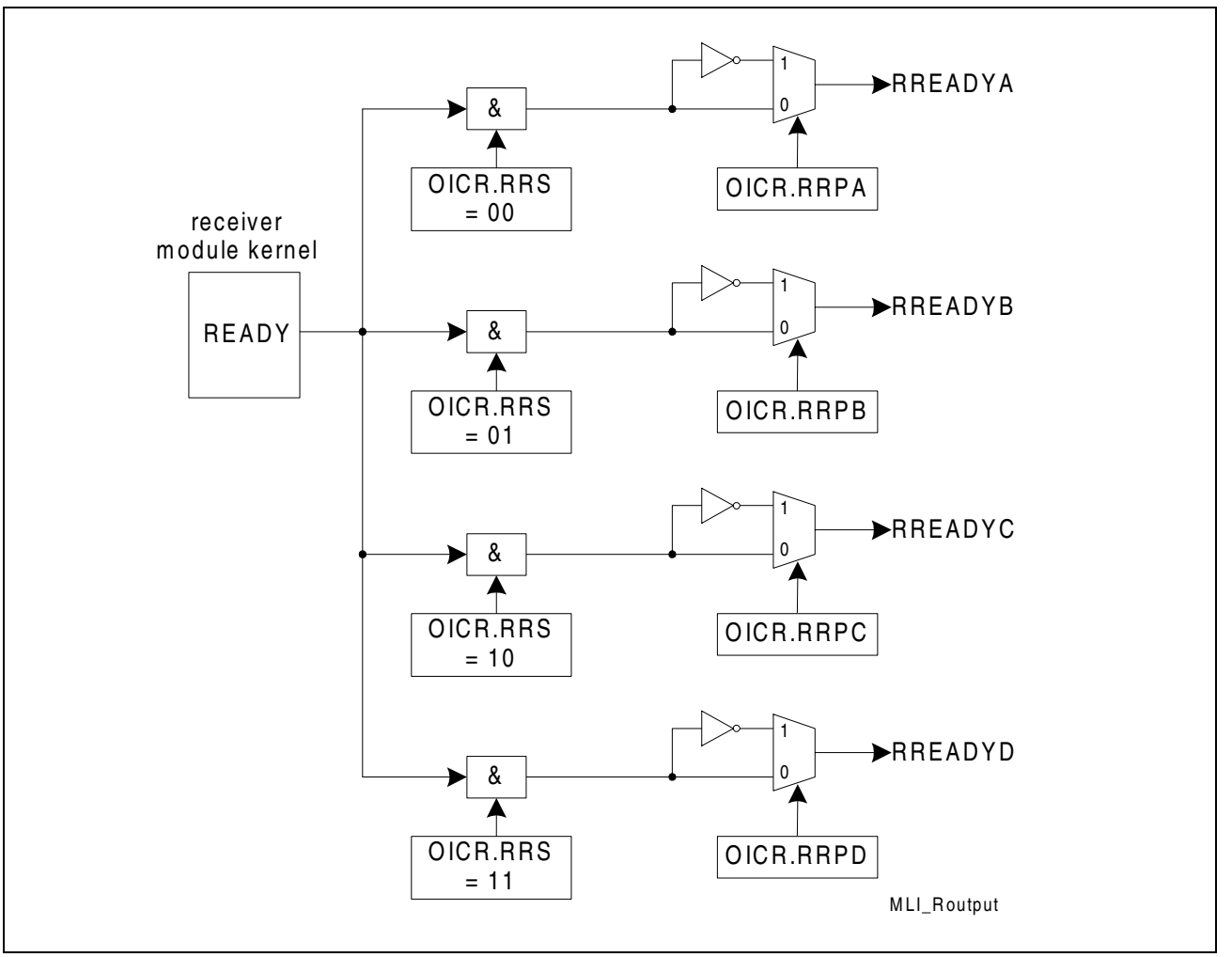


Figure 27-33 Control of Receiver Output Signal

For the CLK and DATA lines, it is possible to choose which of them will be active with the OICR.RCS and OICR.RDS bit fields. Their polarities can be set by programming the bits OICR.RCP and OICR.RDP.

27.1.9 Reading Process Summary

Figure 27-34 illustrates the whole reading process, since it is originated by the first controller, until it gets the answer.

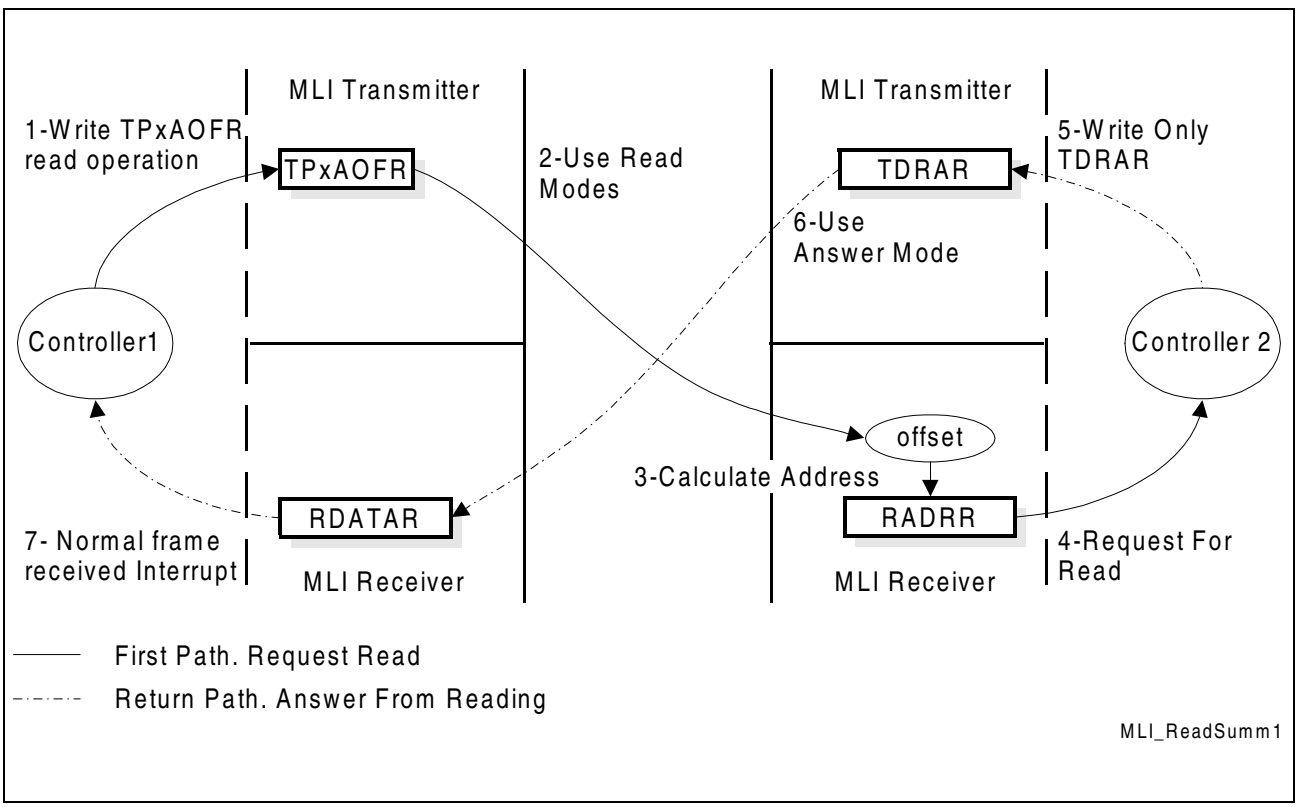


Figure 27-34 Full Read Operation Process

In step 2, the MLI transmitter of controller 1 sets TRSTATR.RPx and TRSTATR.DVx equal to one. Then it sends the read frame (optimized or not). When this frame is properly received, the MLI transmitter resets again the TRSTATR.DVx flag.

In step 3, the MLI receiver of controller 2 calculates the address and writes TSTATR.APN with the value received in the read frame. In step 6, the MLI transmitter of controller 2 sends an answer frame using the pipe number indicated in TSTATR.APN.

In step 7, the MLI receiver of the controller 1 gets an answer frame. If the read pending flag is 0 and its DVx is 1, then the frame is discarded and the MLI receiver produces an interrupt if enabled by RIER.DRAIE. If the RPx is set and DVx is reset, the MLI receiver produces a normal frame received interrupt (if enabled by CIR.NFRIE) to inform its CPU that it has the answer.

The software may read the data width and the address offset from the TPxSTATR TPxAOFR registers (the pipe is indicated by RCR.PN). After this read operation is finished, a next read to the transfer window can be initiated.

27.1.10 MLI Interrupts

The general interrupt structure is shown in **Figure 27-35**. The interrupt event can trigger the interrupt generation and sets the corresponding bit in the status register. The interrupt pulse is generated independently from the interrupt flag in the interrupt status register. The interrupt flag can be reset by software.

If the interrupt is enabled by the related interrupt enable bit in the interrupt enable register, an interrupt pulse can be generated at one of the interrupt output lines INT_Ox of the module. If more than one interrupt source are connected to the same interrupt node pointer (in the interrupt node pointer register), the requests are combined to one common line.

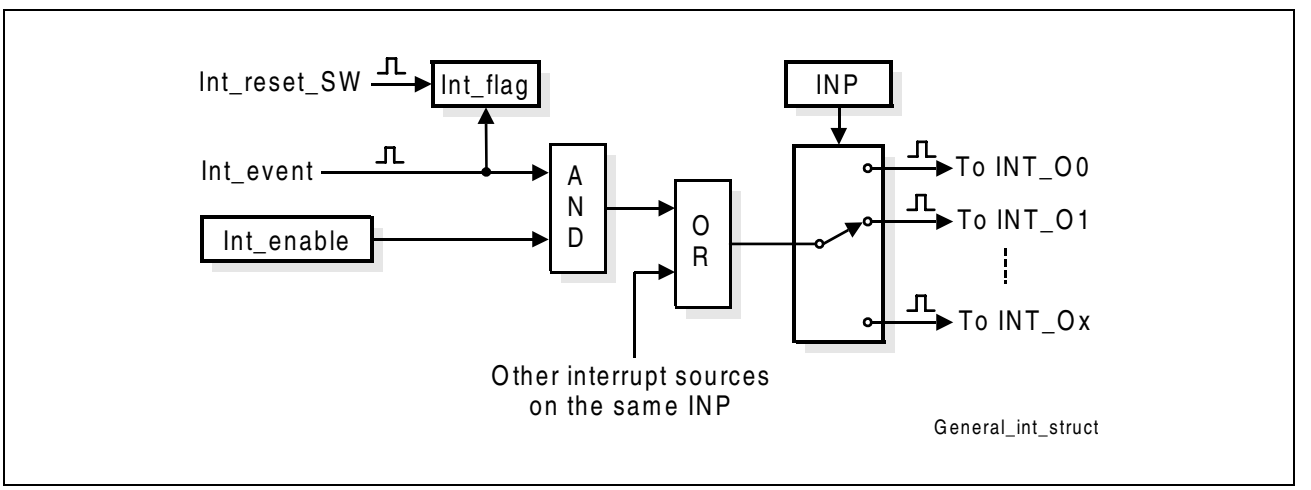


Figure 27-35 General Interrupt Structure

The request compressor condenses the MLI interrupt request sources to 8 interrupt outputs, reporting the interrupt requests of the MLI module to the interrupt controller. Each request source is provided with an interrupt output pointer, selecting the interrupt output to start the associated service routine to increase flexibility in interrupt processing. Each of the 8 interrupt outputs can trigger an independent routine with its own interrupt vector and its own priority.

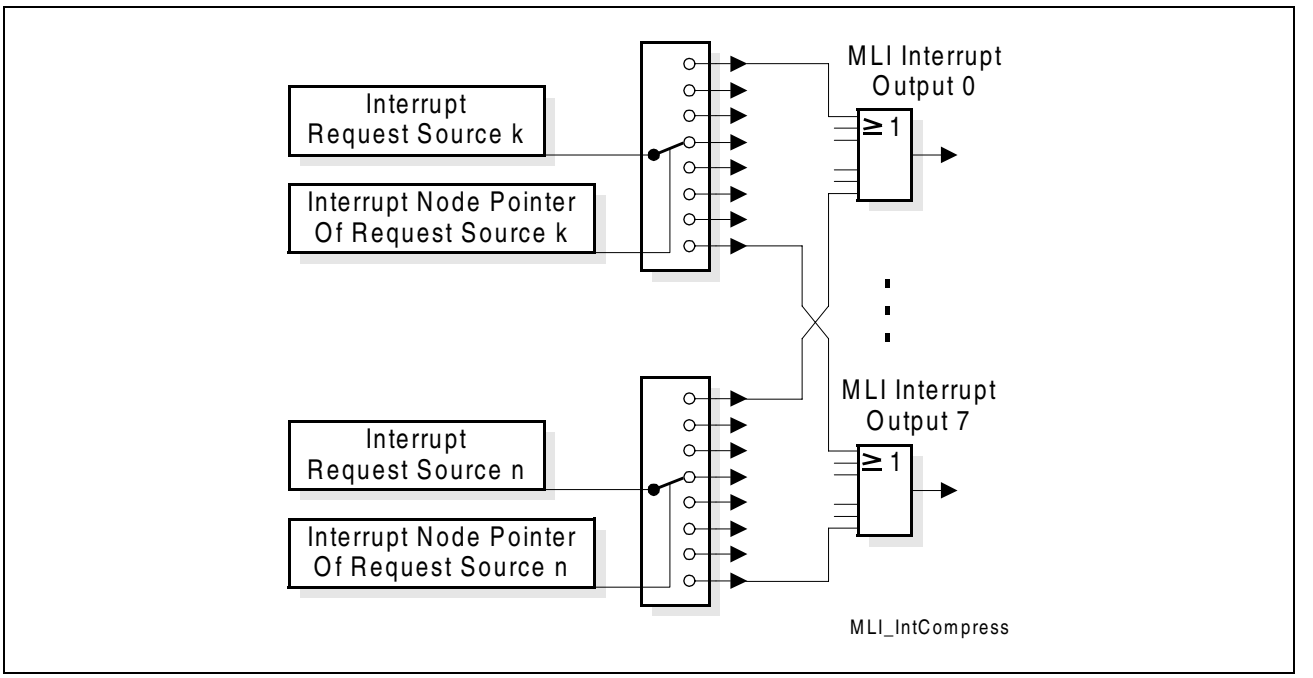


Figure 27-36 Interrupt Output Pointer and Interrupt Request Compressor

Table 27-25 shows the MLI interrupts.

Table 27-25 MLI Interrupts

Description and Condition	Agent that Generates it	Controlled by
Parity error interrupt. Produced when a programmable maximum number of parity errors is reached.	MLI transmitter	TINPR.PTEIP TIER.PEIE TIER.PEIR
Time-out error interrupt. Generated when a programmable maximum number of non acknowledge errors is reached.	MLI transmitter	TINPR.PTEIP TIER.TEIE TIER.TEIR
Normal frame sent interrupt. MLI transmitter has sent a normal frame (not command) through pipe number x.	MLI transmitter	TINPR.NFSIPx TIER.NFSIEx TIER.NFSIRx
Command frame sent interrupt. MLI transmitter has sent a command frame through pipe number x, (only pipes 0, 1, 2 and 3).	MLI transmitter	TINPR.CFSIP TIER.CFSIEx TIER.CFSIRx
Discarded read answer received interrupt. Produced whenever an answer frame is received and the RPx flag of its correspondent pipe is 0, or RPx and DVx are 1.	MLI receiver	RINPR.DRAIP RIER.DRAIE RIER.DRAIR

Table 27-25 MLI Interrupts (cont'd)

Description and Condition	Agent that Generates it	Controlled by
Memory protection or parity error interrupt. Produced when detected a non allowed read or write access or when a programmable maximum number of parity errors is reached.	MLI receiver	RINPR.MPPEIP RIER.MPEIE RIER.PEIE RIER.MPEIR RIER.PEIR
Normal frame received interrupt. The MLI receiver has obtained a normal frame (not command).	MLI receiver	RINPR.NFRIP RIER.NFRIE RIER.NFRIR
Command frame received interrupt. The MLI receiver has obtained a command frame through pipe number x.	MLI receiver	RINPR.CFRIP RIER.CFRIEx RIER.CFRIRx

27.1.11 Clock Domains and Handshake Timing

Figure 27-37 illustrates how the signals are synchronized in the MLI transmitter and in the receiver. In the figure, the suffixes T1 and R2 indicate from which side the signals are seen (from the MLI1 transmitter or from the MLI2 receiver).

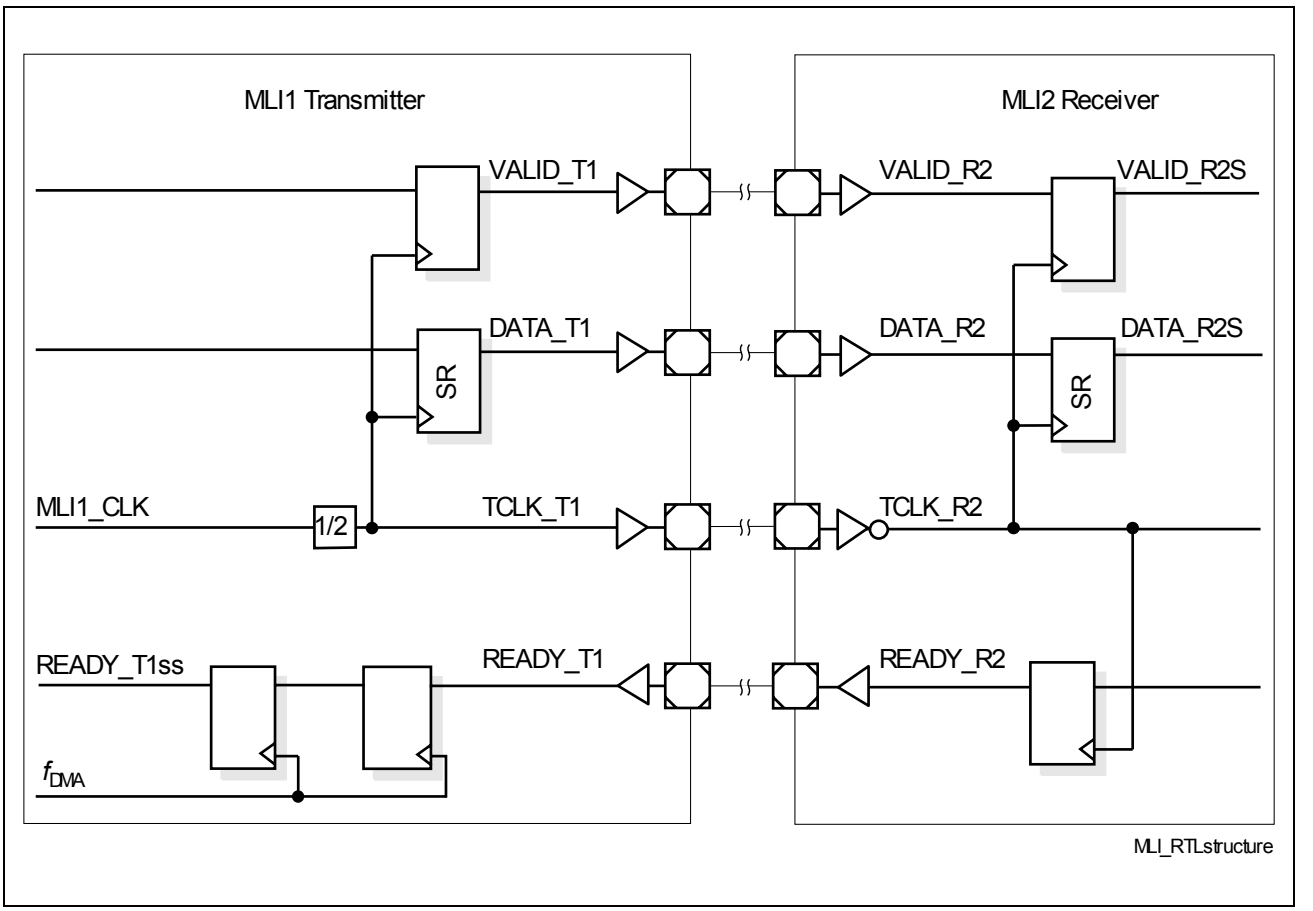


Figure 27-37 Signals Synchronization in MLI Transmitter and Receiver

Note: In the figure above SR stands for shift register.

The physical connection between both MLI will introduce a delay that will be dependent of the concrete application and must be characterized in order to choose proper values of MDP and RCR.DPE.

Figure 27-38 shows a detailed time diagram of a correct transfer. Each of the signals is shown in **Figure 27-37**. The delay d1 is the one that affects the signals coming out from the MLI transmitter and d2 is the delay associated to the READY signal.

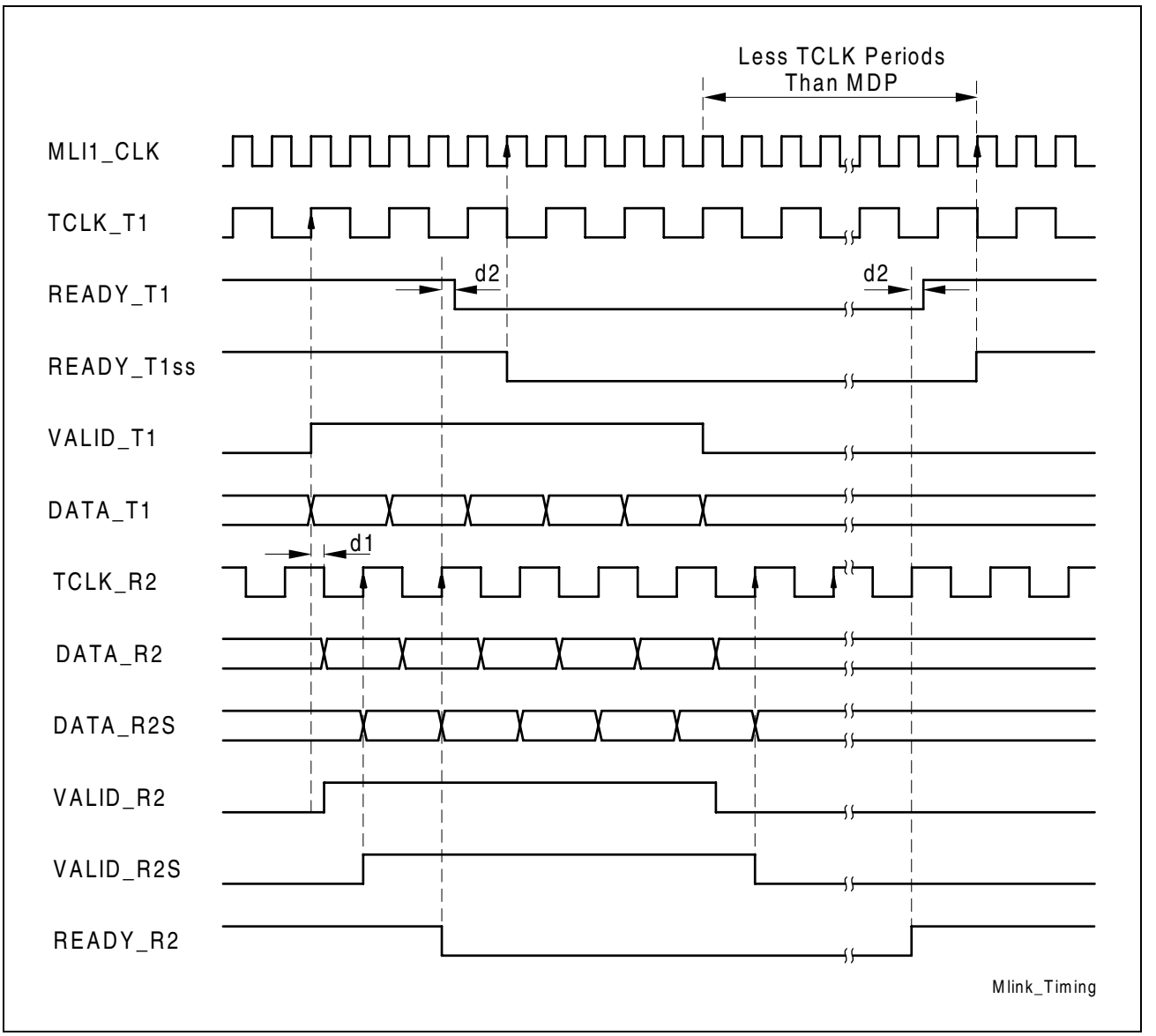


Figure 27-38 Detailed Handshake in a Correct Transfer

Figure 27-39 illustrates how the MLI receiver informs the transmitter that it has received a frame with a parity error.

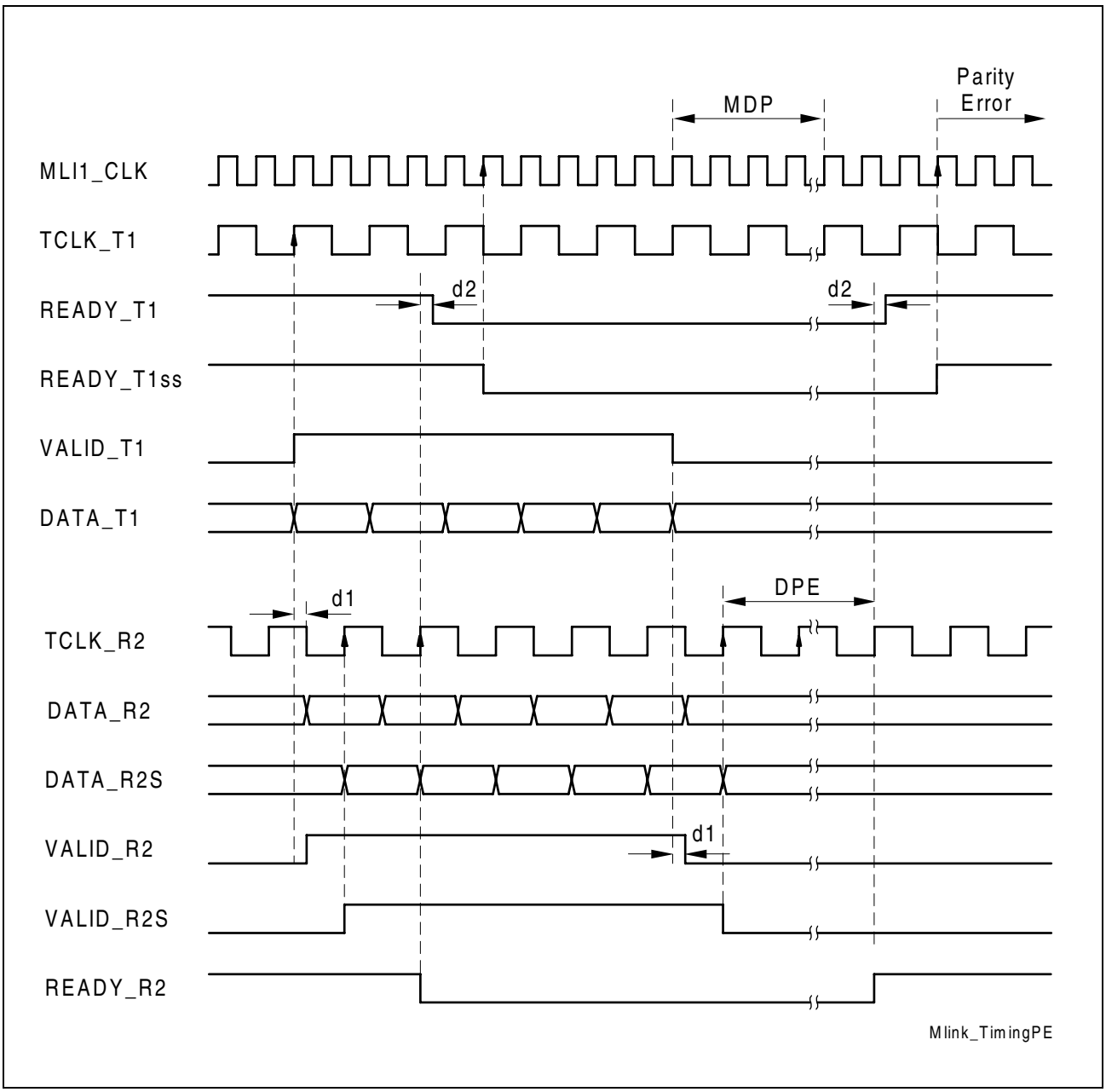


Figure 27-39 Detailed Handshake in a Transfer with Parity Error

Figure 27-40 illustrates the situation in which the MLI receiver has not acknowledged a transmission.

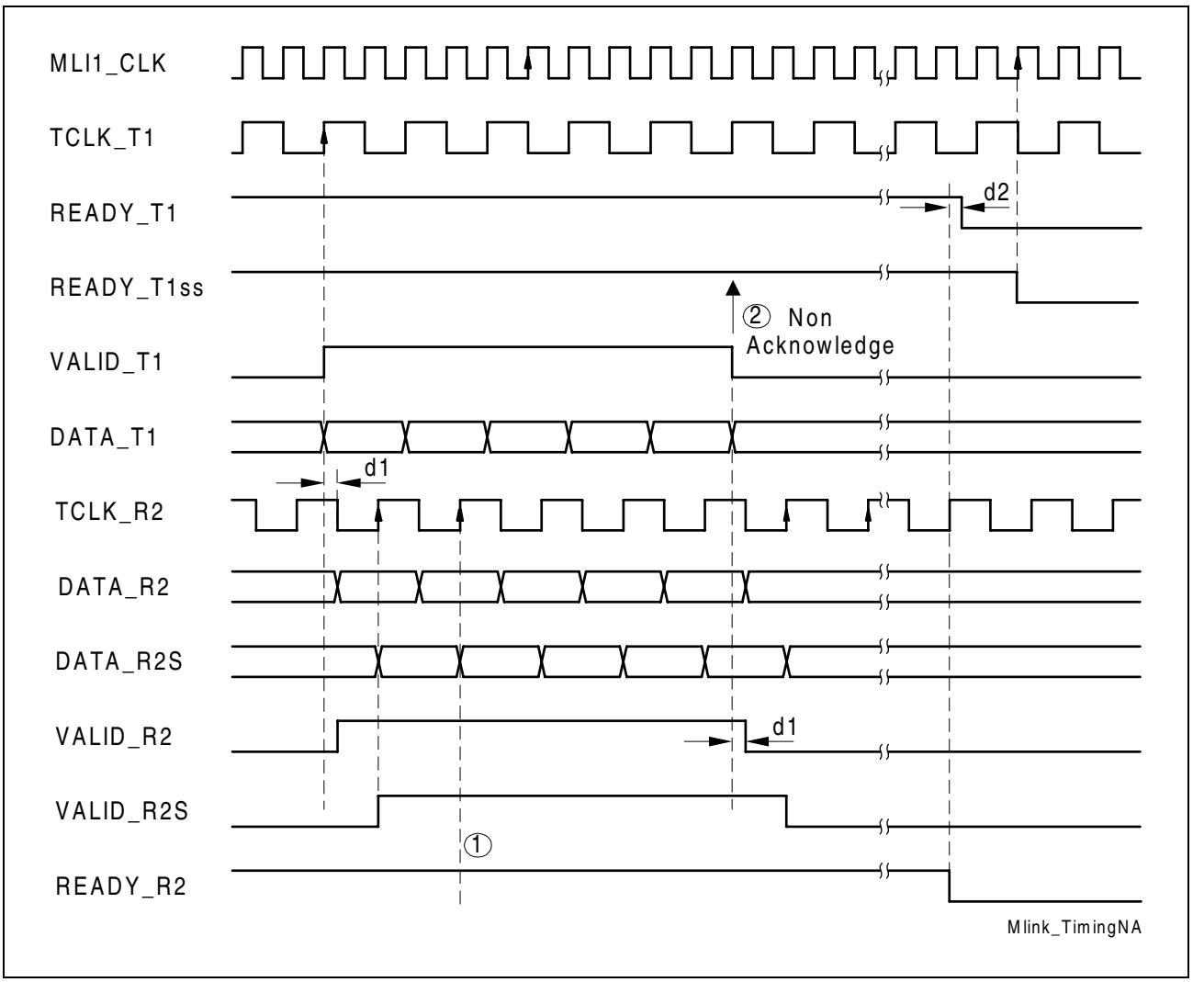


Figure 27-40 Detailed Handshake in a Transfer with Non Acknowledge Error

In **Figure 27-40**, number 1 represents the moment in which the MLI receiver should set its READY signal. Number 2 represents the instant in which the MLI transmitter checks the READY signal status.

A non acknowledge situation may lead into a time-out if the MLI receiver does not raise again the READY signal before the counter of non acknowledge errors overflow.

27.1.12 Data Flow Description

27.1.12.1 Copy Base Address

The copy base address frame is used to transmit the two parameters of a remote window for pipe x, the 28 most significant base address bits and the 4-bit coded buffer size, from the local microcontroller to the remote microcontroller.

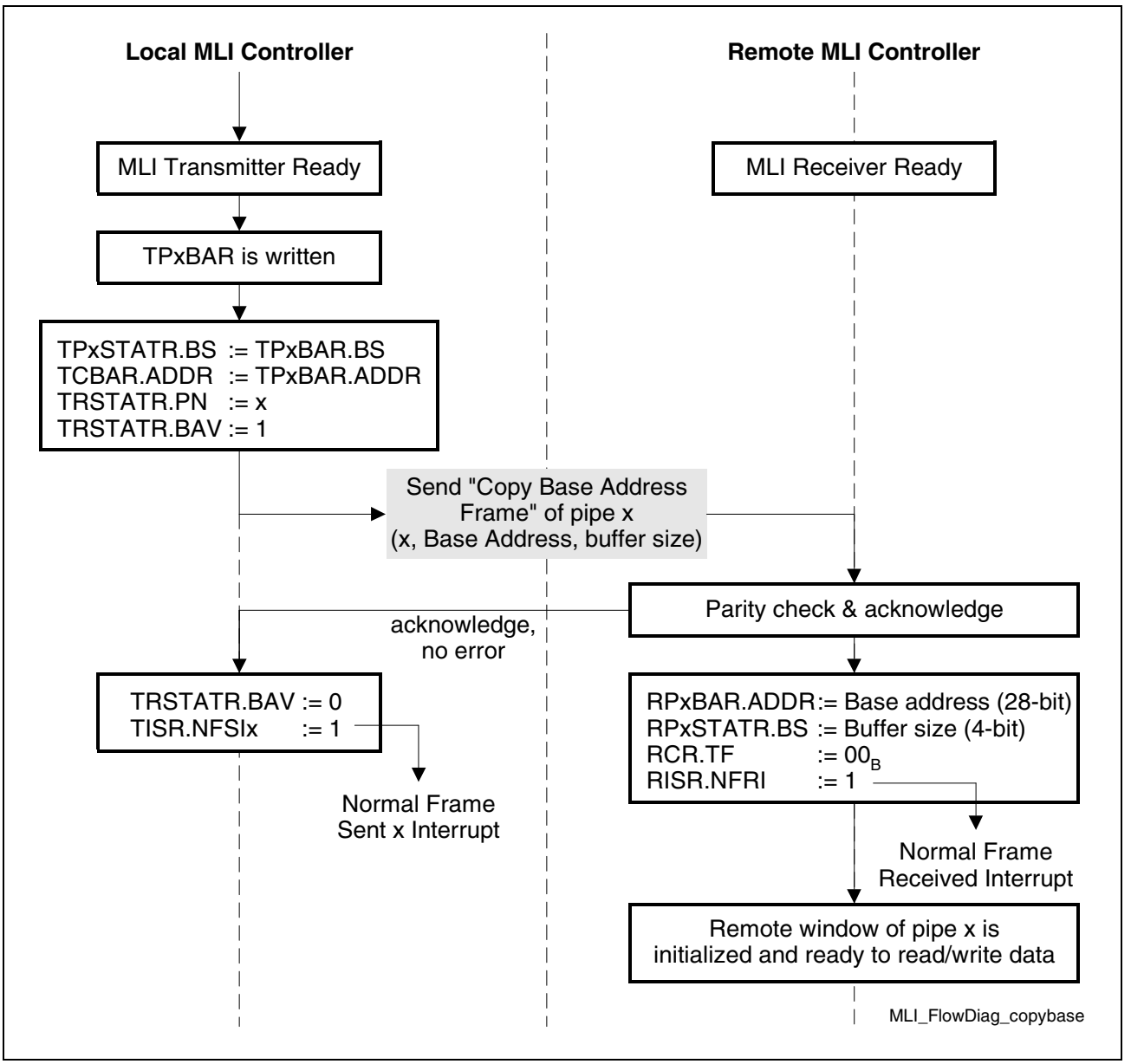

Local MLI Controller
Remote MLI Controller
MLI Transmitter Ready
MLI Receiver Ready
TPxBAR is written
TPxSTATR.BS := TPxBAR.BS
TCBAR.ADDR := TPxBAR.ADDR
TRSTATR.PN := x
TRSTATR.BAV := 1
Send "Copy Base Address Frame" of pipe x (x, Base Address, buffer size)
Parity check & acknowledge
acknowledge, no error
TRSTATR.BAV := 0
TISR.NFSIx := 1
RPxBAR.ADDR := Base address (28-bit)
RPxSTATR.BS := Buffer size (4-bit)
RCR.TF := 00B
RISR.NFRI := 1
Normal Frame Sent x Interrupt
Normal Frame Received Interrupt
Remote window of pipe x is initialized and ready to read/write data
MLI_FlowDiag_copybase

Figure 27-41 Copy Base Address Frame Flow

The transmission of a copy base address frame is initiated by writing the two parameters for a pipe remote window, the 28 most significant base address bits and the 4-bit coded remote window (buffer) size, into register TPxBAR.

27.1.12.2 Command Frame

The transmission of a command frame is initiated by writing one of the four pipe x related command code bit fields in register TCMDR. Depending on the pipe x related command code that is transmitted, different actions are triggered in the remote controller.

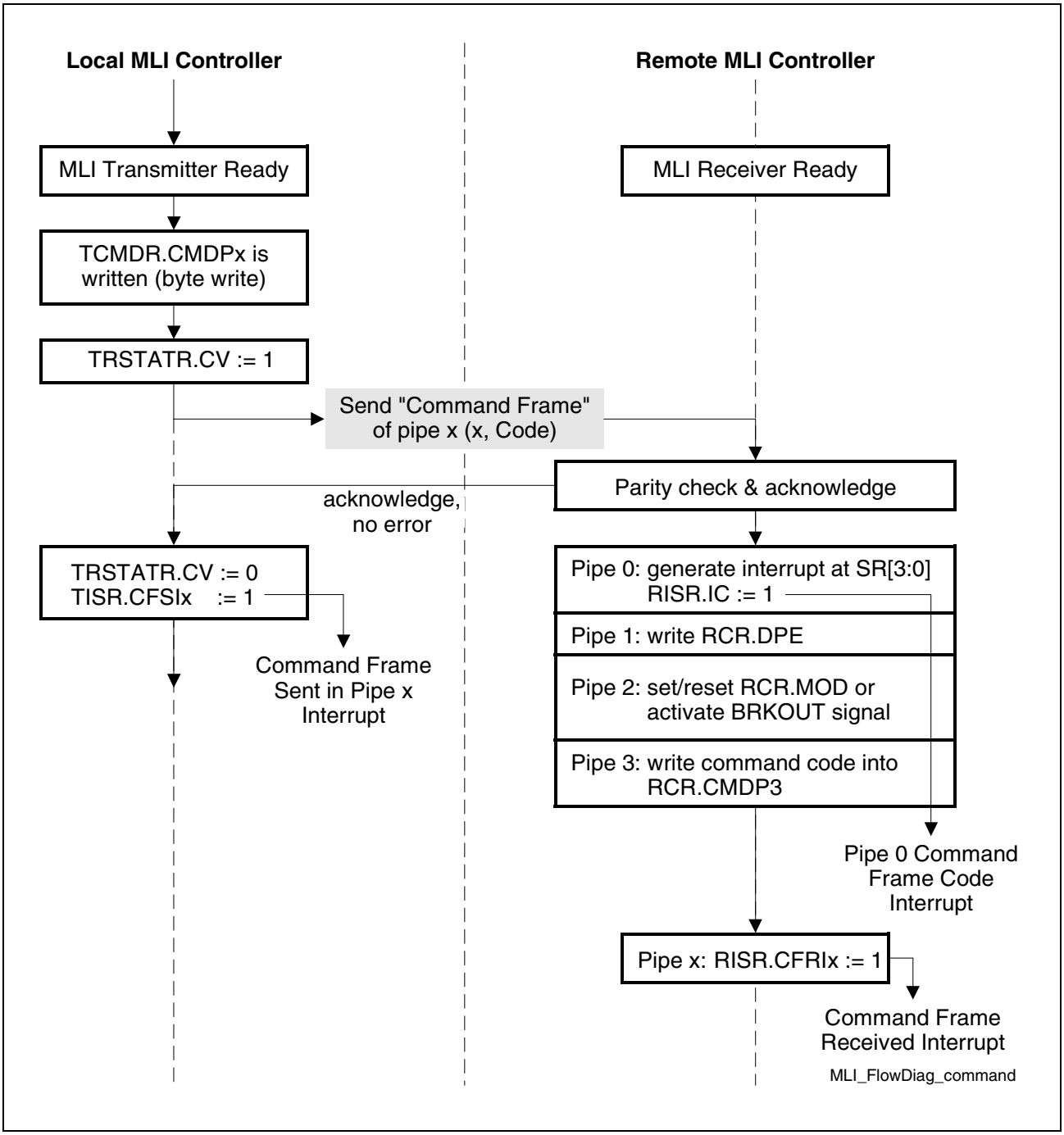

Local MLI Controller
Remote MLI Controller
MLI Transmitter Ready
MLI Receiver Ready
TCMDR.CMDPx is written (byte write)
TRSTATR.CV := 1
Send "Command Frame" of pipe x (x, Code)
Parity check & acknowledge
acknowledge, no error
TRSTATR.CV := 0
TISR.CFSIx := 1
Command Frame Sent in Pipe x Interrupt
Pipe 0: generate interrupt at SR[3:0]
RISR.IC := 1
Pipe 1: write RCR.DPE
Pipe 2: set/reset RCR.MOD or activate BRKOUT signal
Pipe 3: write command code into RCR.CMDP3
Pipe 0 Command Frame Code Interrupt
Pipe x: RISR.CFRIx := 1
Command Frame Received Interrupt
MLI_FlowDiag_command

Figure 27-42 Command Frame Flow

27.1.12.3 Write Frame

The transmission of a write frame is initiated in the local controller by a write access of a bus master (e.g. the CPU) to one of the four transfer windows.

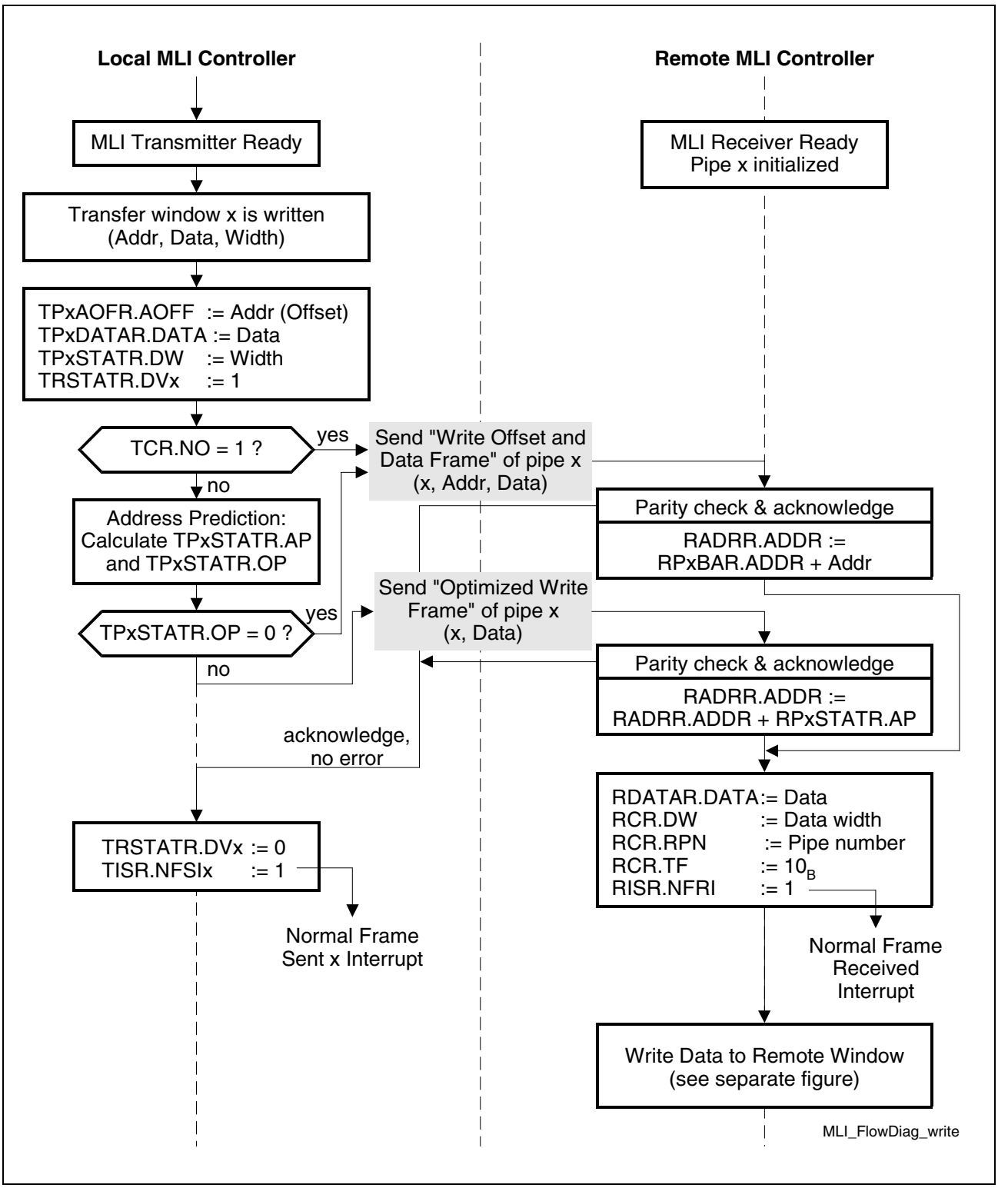


Figure 27-43 Write Frame Flow

27.1.12.4 Read Frame

The transmission of a read frame (see **Figure 27-44**) is initiated in the local controller by a read access of a bus master (e.g. the CPU) to one of the four transfer windows.

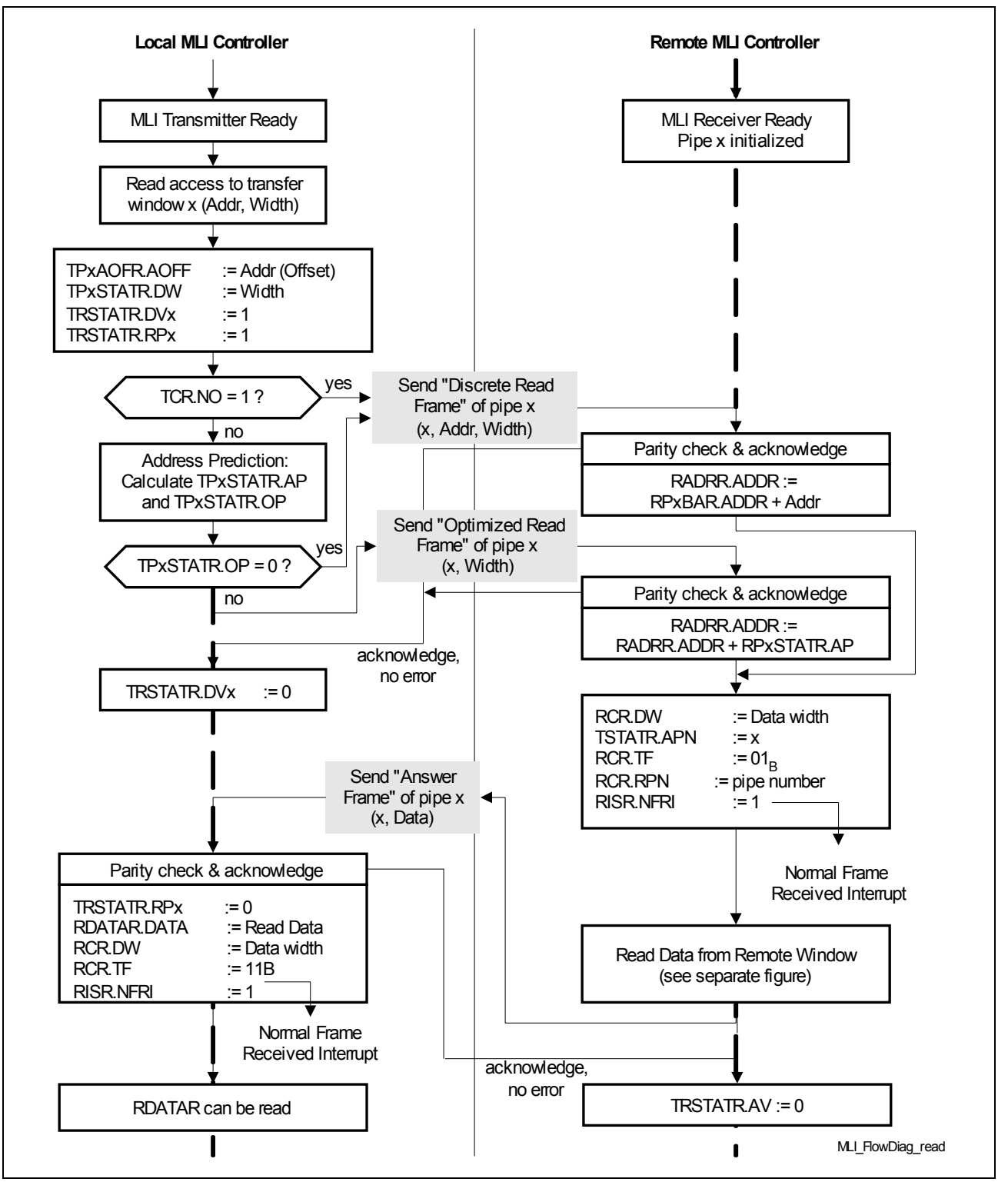


Figure 27-44 Read Frame Flow

27.1.12.5 Access to Remote Window

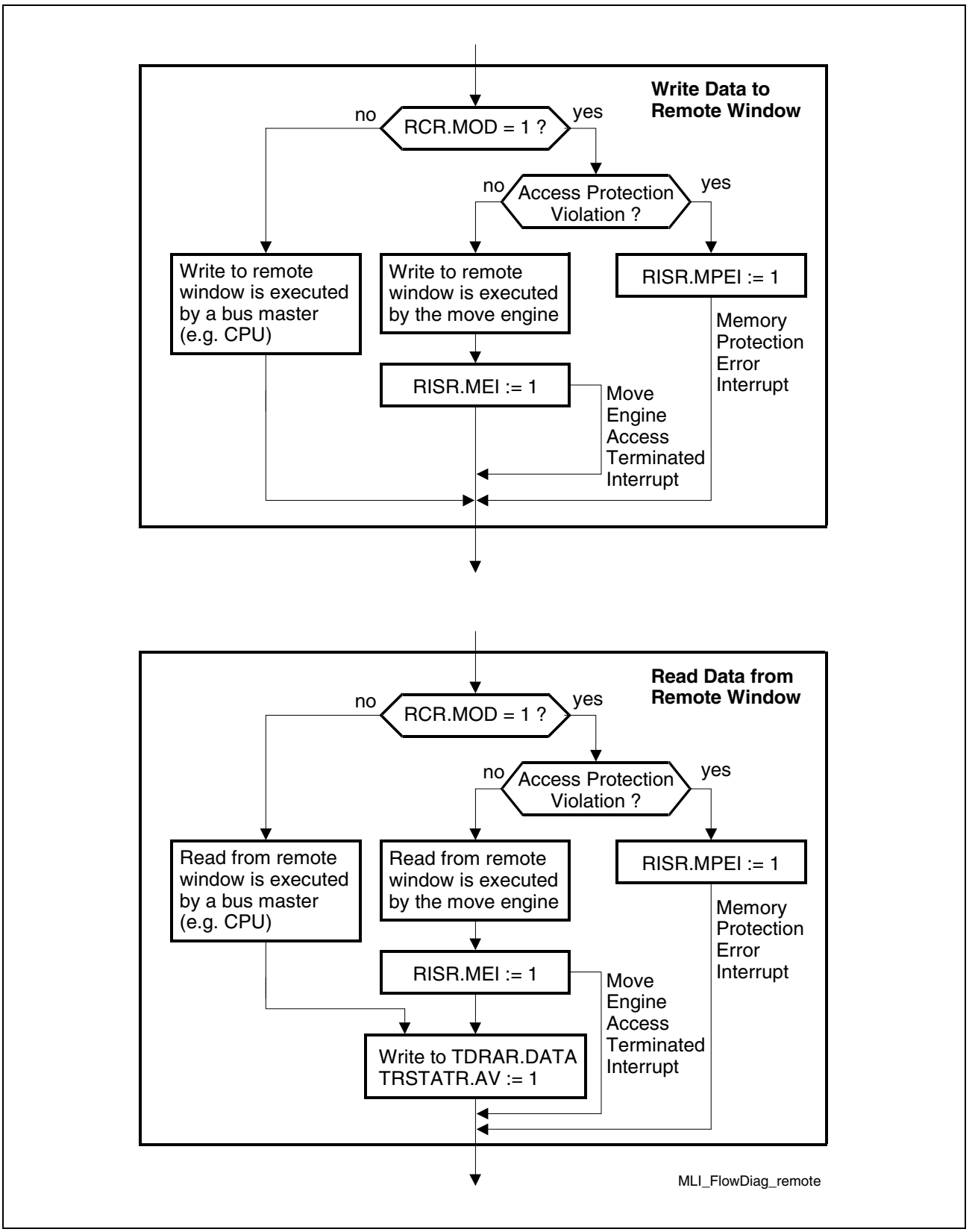

Write Data to Remote Window
RCR.MOD = 1 ?
no
yes
Access Protection Violation ?
no
yes
Write to remote window is executed by a bus master (e.g. CPU)
Write to remote window is executed by the move engine
RISR.MPEI := 1
RISR.MEI := 1
Memory Protection Error Interrupt
Move Engine Access Terminated Interrupt
Read Data from Remote Window
RCR.MOD = 1 ?
no
yes
Access Protection Violation ?
no
yes
Read from remote window is executed by a bus master (e.g. CPU)
Read from remote window is executed by the move engine
RISR.MPEI := 1
RISR.MEI := 1
Memory Protection Error Interrupt
Move Engine Access Terminated Interrupt
Write to TDRAR.DATA TRSTATR.AV := 1
MLI_FlowDiag_remote

Figure 27-45 Access to Remote Window

27.2 MLI Kernel Registers

Figure 27-46 and **Table 27-26** show all registers associated with the MLI Kernel.

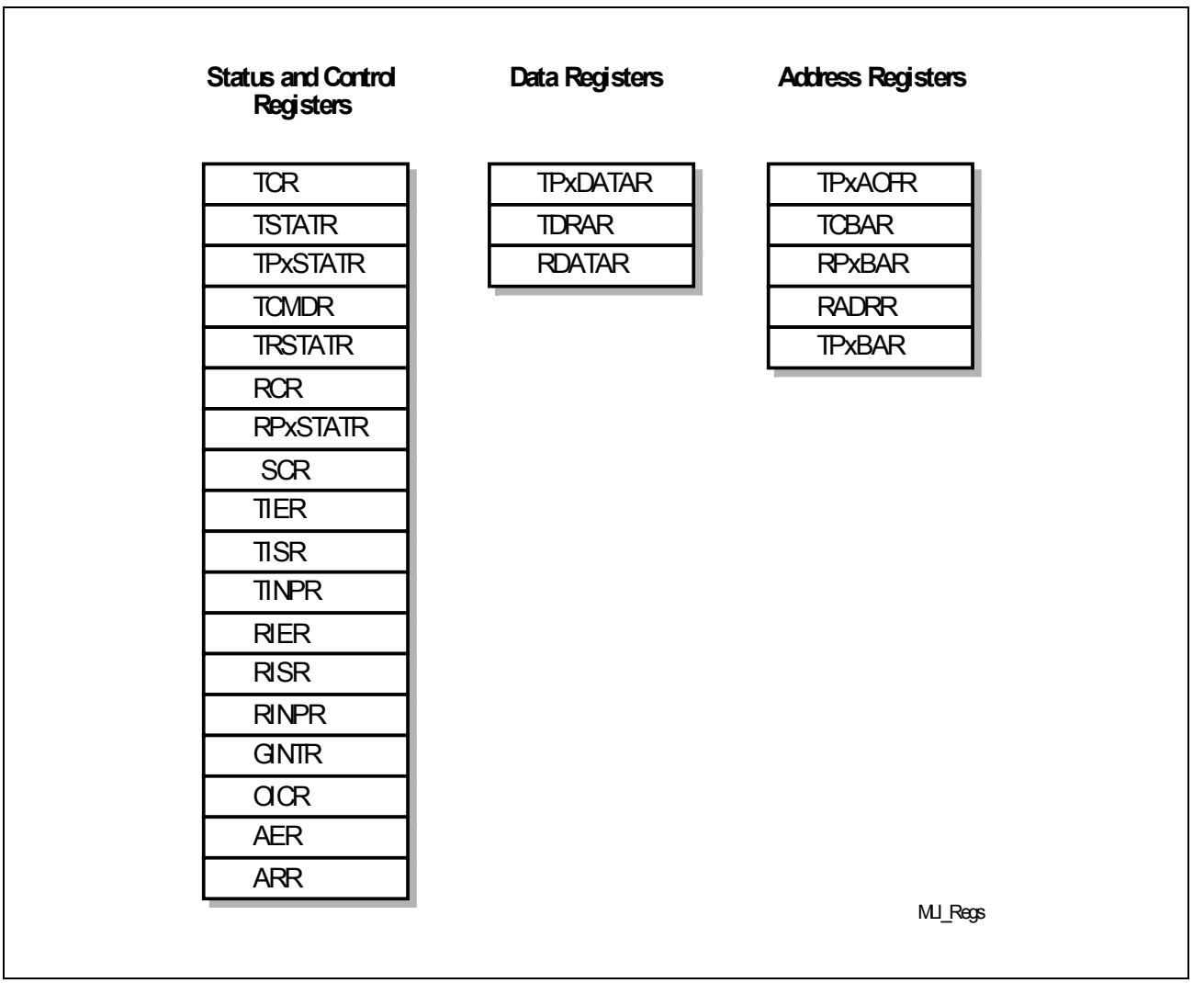


Figure 27-46 MLI Kernel Registers

Note: The letter “x” indicates the number of pipe (pipes 0, 1, 2, and 3).

Note: All bits marked ‘w’ return 0 when read.

Table 27-26 MLI Kernel Registers

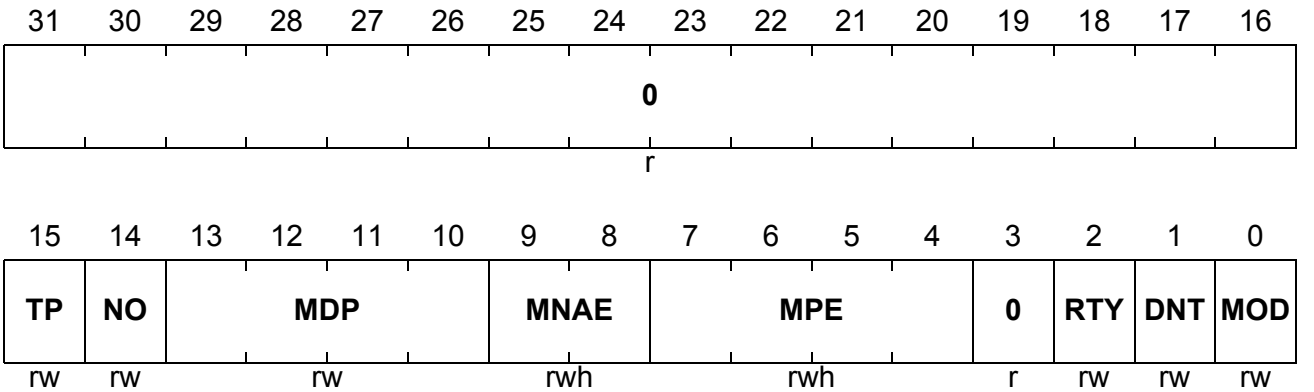
Register Short Name	Register Long Name	Offset Address	Description see
TCR	Transmitter Control Register	0010_H	Page 27-62
TSTATR	Transmitter Status Register	0014_H	Page 27-64
TP0STATR	Transmitter Pipe 0 Status Register	0018_H	Page 27-66
TP1STATR	Transmitter Pipe 1 Status Register	$001C_H$	Page 27-66
TP2STATR	Transmitter Pipe 2 Status Register	0020_H	Page 27-66
TP3STATR	Transmitter Pipe 3 Status Register	0024_H	Page 27-66
TCMDR	Transmitter Command Register	0028_H	Page 27-67
TRSTATR	Transmitter Registers Status Register	$002C_H$	Page 27-69
TP0AOFR	Transmitter Pipe 0 Address Offset Register	0030_H	Page 27-71
TP1AOFR	Transmitter Pipe 1 Address Offset Register	0034_H	Page 27-71
TP2AOFR	Transmitter Pipe 2 Address Offset Register	0038_H	Page 27-71
TP3AOFR	Transmitter Pipe 3 Address Offset Register	$003C_H$	Page 27-71
TP0DATAR	Transmitter Pipe 0 Data Register	0040_H	Page 27-71
TP1DATAR	Transmitter Pipe 1 Data Register	0044_H	Page 27-71
TP2DATAR	Transmitter Pipe 2 Data Register	0048_H	Page 27-71
TP3DATAR	Transmitter Pipe 3 Data Register	$004C_H$	Page 27-71
TDRAR	Transmitter Data Read Answer Register	0050_H	Page 27-72
TP0BAR	Transmitter Pipe 0 Base Address Register	0054_H	Page 27-72
TP1BAR	Transmitter Pipe 1 Base Address Register	0058_H	Page 27-72
TP2BAR	Transmitter Pipe 2 Base Address Register	$005C_H$	Page 27-72
TP3BAR	Transmitter Pipe 3 Base Address Register	0060_H	Page 27-72
TCBAR	Transmitter Copy Base Address Register	0064_H	Page 27-73
RCR	Receiver Control Register	0068_H	Page 27-74
RP0BAR	Receiver Pipe 0 Base Address Register	$006C_H$	Page 27-77
RP1BAR	Receiver Pipe 1 Base Address Register	0070_H	Page 27-77
RP2BAR	Receiver Pipe 2 Base Address Register	0074_H	Page 27-77
RP3BAR	Receiver Pipe 3 Base Address Register	0078_H	Page 27-77
RP0STATR	Receiver Pipe 0 Status Register	$007C_H$	Page 27-78
RP1STATR	Receiver Pipe 1 Status Register	0080_H	Page 27-78
RP2STATR	Receiver Pipe 2 Status Register	0084_H	Page 27-78

Table 27-26 MLI Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
RP3STATR	Receiver Pipe 3 Status Register	0088_H	Page 27-78
RADRR	Receiver Address Register	$008C_H$	Page 27-79
RDATAR	Receiver Data Register	0090_H	Page 27-79
SCR	Set Clear Register	0094_H	Page 27-80
TIER	Transmitter Interrupt Enable Register	0098_H	Page 27-86
TISR	Transmitter Interrupt Status Register	$009C_H$	Page 27-87
TINPR	Transmitter Interrupt Node Pointer Register	$00A0_H$	Page 27-88
RIER	Receiver Interrupt Enable Register	$00A4_H$	Page 27-90
RISR	Receiver Interrupt Status Register	$00A8_H$	Page 27-92
RINPR	Receiver Interrupt Node Pointer Register	$00AC_H$	Page 27-93
GINTR	Global Interrupt Set Register	$00B0_H$	Page 27-94
OICR	Output Input Control Register	$00B4_H$	Page 27-81
AER	Access Enable Register	$00B8_H$	Page 27-95
ARR	Access Range Register	$00BC_H$	Page 27-96

27.2.1 MLI Transmitter Registers

TCR
Transmitter Control Register **Reset Value: 0000 0110$_H$**

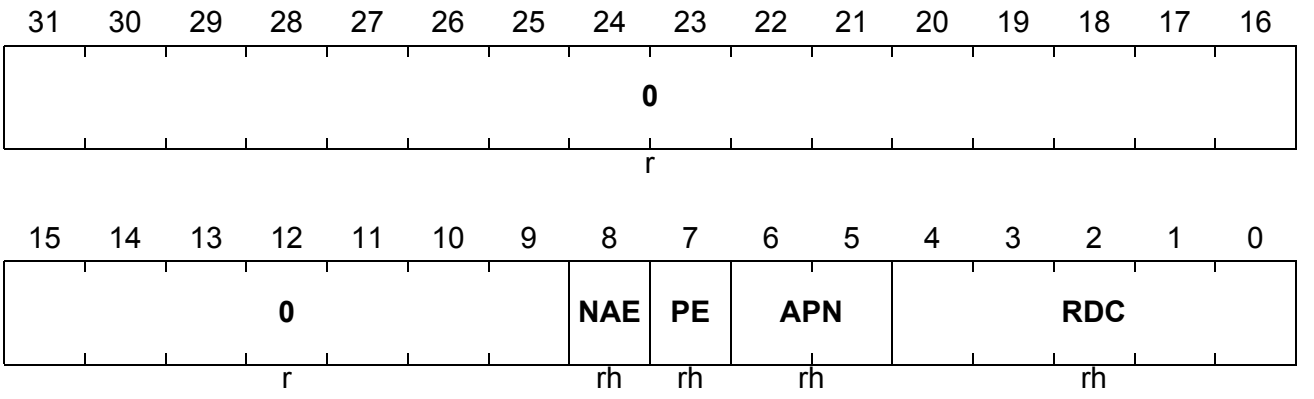


Field	Bits	Type	Description
MOD	0	rw	**Mode of Operation** This bit establishes the operation mode of the MLI transmitter. Its encoding is as follows: 0 MLI transmitter off 1 MLI transmitter on
DNT	1	rw	**Data in Not Transmission** This bit will determine the level of the data line when no transmission is in progress. If set to one, the data line when the transmission is finished will have the value 1. If set to zero the DATA line level will be 0.
RTY	2	rw	**Retry** This bit enables the retry mechanism for the transfer windows. 0 The retry mechanism is disabled. Any access while the transmitter is busy is discarded without additional action. 1 The retry mechanism is enabled. Any access while the transmitter is busy is acknowledged with a retry. In this case, the requesting bus master sends the requested access again until the request is accepted.

Field	Bits	Type	Description
MPE	[7:4]	rwh	**Maximum Parity Errors** This bit field indicates the value of parity errors for the parity interrupt generation. It is set to its maximum value by software and its number will be decreased by the MLI, each time it detects a parity error. Its encoding is as follows: 0000_B Generate parity error interrupt 0001_B 1 parity error for interrupt generation 0010_B 2 parity errors for interrupt generation … 1111_B 15 parity errors for interrupt generation
MNAE	[9:8]	rwh	**Maximum Non Acknowledge Errors** This bit field indicates the number of acknowledge errors for the time out interrupt generation. It is set to its maximum value (11_B) by software and its number will be decreased by the MLI, each time it detects a non acknowledge error. Its encoding is as follows: 00_B Generate non acknowledge error interrupt when acknowledge error occurs 01_B One error for the interrupt generation; Interrupt will be generated when acknowledge error occurs 10_B Two errors for the interrupt generation 11_B Three errors for the interrupt generation If the non acknowledge error does not appear again before the counter reaches zero, the MLI sets again this field to 11_B.
MDP	[13:10]	rw	**Maximum Delay for Parity Error** This bit field is written by software and it defines the number of clock periods (from the clock used in the transmission) above which if the READY signal remains low it will be considered parity error condition. These bits will be interpreted as follows: 0000_B Zero clock periods 0001_B One clock period … 1110_B Fourteen clock periods 1111_B Fifteen clock periods

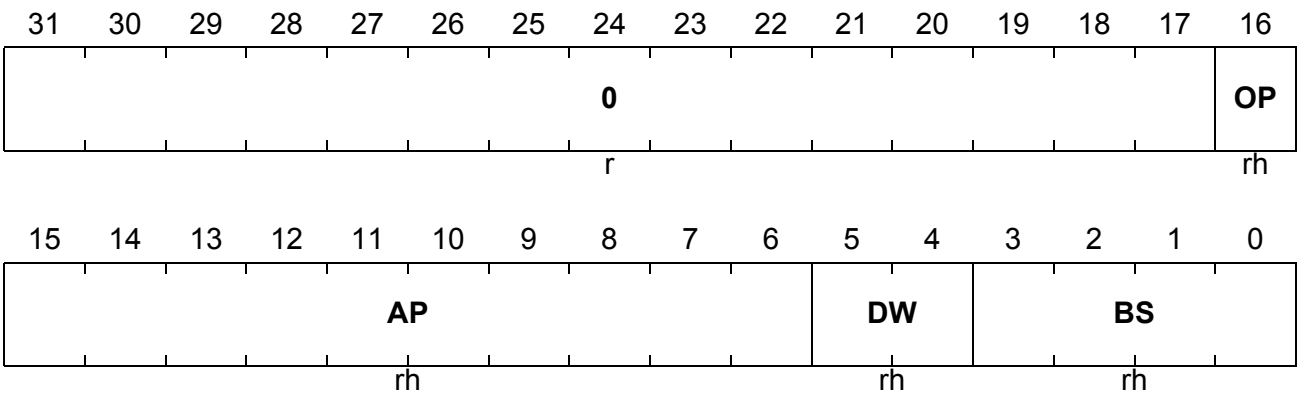
Field	Bits	Type	Description
NO	14	rw	**No Optimized Method** This bit field indicates if the optimized method for address prediction is enabled or not. Its encoding is as follows: 0 Optimized method enabled 1 Optimized method not enabled
TP	15	rw	**Type of Parity** This bit will determine the type of parity used in the transmission. This value will be the one initially used for the toggle bit that produces the parity bit. Assuming a correct transmission, when set to one it will force the MLI receiver to produce a parity error condition.
0	3, [31:16]	r	**Reserved;** read as 0; should be written with 0.

TSTATR
Transmitter Status Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RDC	[4:0]	rh	**Ready Delay Counter** This counter is reset to zero when the VALID signal goes low level after a transmission, and it will count TCLK periods until the MLI transmitter detects that the READY signal is high level again, or when the counter reaches its maximum value.
APN	[6:5]	rh	**Answer Pipe Number** This bit field is written by the MLI receiver whenever it gets a read frame. This value will be coincident with the pipe number of the received read frame and it will be used to send the answer frame. Its encoding is as follows: 00_B Send the answer frame through pipe 0 01_B Send the answer frame through pipe 1 10_B Send the answer frame through pipe 2 10_B Send the answer frame through pipe 3
PE	7	rh	**Parity Error Flag** Set to one when the MLI transmitter detects a parity error in the transmission. It is reset again when the MLI makes a transfer without parity error or when set the SCR.CTPE bit.
NAE	8	rh	**Non Acknowledge Error Flag** Set to one when the MLI transmitter detects a non acknowledge error in the transmission. It is reset again when the MLI makes a transfer without error or when set the SCR.CNAE bit.
0	[31:9]	r	**Reserved;** read as 0; should be written with 0.

TP0STATR
Transmitter Pipe 0 Status Register **Reset Value: 0000 0000$_H$**
TP1STATR
Transmitter Pipe 1 Status Register **Reset Value: 0000 0000$_H$**
TP2STATR
Transmitter Pipe 2 Status Register **Reset Value: 0000 0000$_H$**
TP3STATR
Transmitter Pipe 3 Status Register **Reset Value: 0000 0000$_H$**

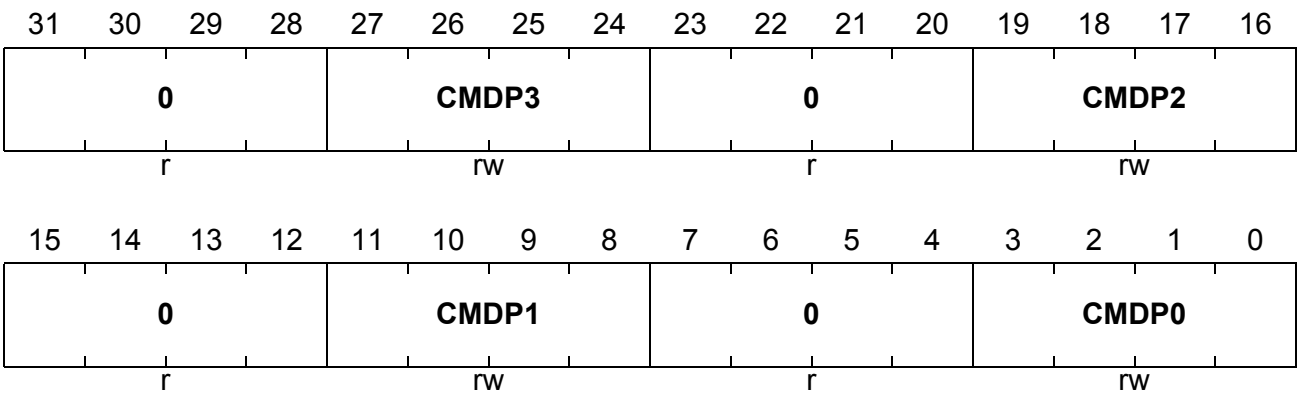


Field	Bits	Type	Description
BS	[3:0]	rh	**Buffer Size** It gives the size of the transfer window in the second controller. The offset width will be coincident with this buffer size field. This field is updated by the MLI transmitter when the correspondent TPxBAR register is written. 0000_B One bit offset 0001_B Two bits offset … 1111_B Sixteen bits offset
DW	[5:4]	rh	**Data Width** This bit field defines the width of the data written in the TPxDATAR register. It is written by the MLI transmitter each time a new data is received in the TPxDATAR register. 00_B Data width of 8 bits selected 01_B Data width of 16 bits selected 10_B Data width of 32 bits selected 11_B Reserved

Field	Bits	Type	Description
AP	[15:6]	rh	**Address Prediction Factor** It is written by the MLI transmitter. It is used to keep track of the address prediction method. It represents a number of ten bits with sign in two's complement.
OP	16	rh	**Use Optimized Frame** This bit field is written by the MLI transmitter each time it performs the necessary operations to know if the address offset follows the address prediction scheme. Its meaning is as follows: 0 Do not use optimized mode to send the frame 1 Use optimized mode to send the frame It is used by the MLI to know if the optimized method should or not be used when a frame is selected to be sent.
0	[31:17]	r	**Reserved;** read as 0; should be written with 0.

The TCMDR register must be written only bite-wisely. Write accesses with a larger data width than a byte are forbidden.

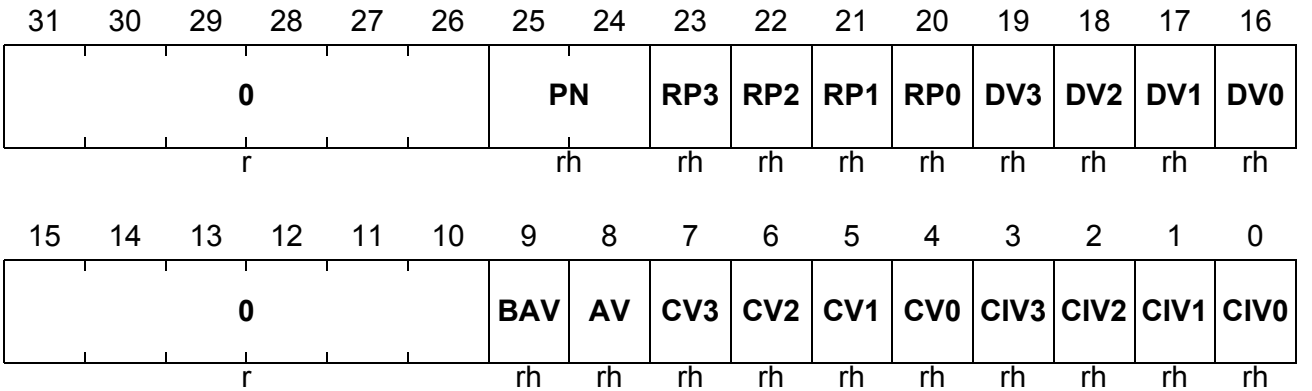
TCMDR
Transmitter Command Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CMDP0	[3:0]	rw	**Command in Pipe 0** This bit field is written by software and it defines the command to be sent through pipe 0. These bits will be interpreted as follows: 0001_B Program MLI receiver to produce interrupt 0 0010_B Program MLI receiver to produce interrupt 1 0011_B Program MLI receiver to produce interrupt 2 0100_B Program MLI receiver to produce interrupt 3 others No effect
CMDP1	[11:8]	rw	**Command in Pipe 1** This bit field is written by software and it defines the command to be sent through pipe 1. These bits will be interpreted as follows: 0000_B Make RCR.DPE = 0000_B 0001_B Make RCR.DPE = 0001_B … 1111_B Make RCR.DPE = 1111_B
CMDP2	[19:16]	rw	**Command in Pipe 2** This bit field is written by software and it defines the command to be sent through pipe 2, see **Table 27-22**.
CMDP3	[27:24]	rw	**Command in Pipe 3** This bit field is written by software and it defines the command to be sent through pipe 3. The commands will be software interpreted.
0	[7:4] [15:12] [23:20] [31:28]	r	**Reserved;** read as 0; should be written with 0.

TRSTATR
Transmitter Registers Status Register

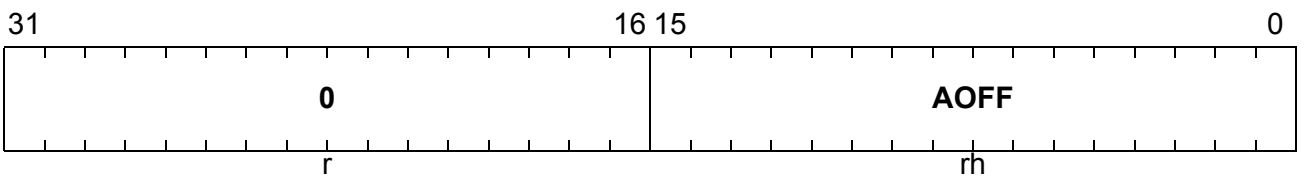
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
CIVx, x = 0, 1, 2, 3	[3:0]	rh	**Command Interrupt Valid** Set to one by the MLI transmitter whenever it detects a rising edge in the correspondent trigger_commandx line. It is reset again when the correspondent command frame to the line is correctly sent through pipe 0, or when set SCR.CCIVx.
CVx, x = 0, 1, 2, 3	[7:4]	rh	**Command Valid** Set to one by the MLI transmitter when the TCMDR.CMDPx bit field is written, or when set SCR.SCVx. It is reset again when the command frame is correctly sent, or when set SCR.CCVx.
AV	8	rh	**Answer Valid** Set to one by the MLI transmitter when the TDRAR register is written. It is reset again when the answer frame is correctly sent, or when set SCR.CAV.
BAV	9	rh	**Base Address Valid** Set to one by the MLI transmitter when the TCBAR register is written. It is reset again when the copy base address frame is correctly sent, or when set SCR.CBAV.
DVx, x = 0, 1, 2, 3	[19:16]	rh	**Data Valid** Set to one by the MLI transmitter when the TPxDATAR and/or the TPxAOFR registers are written. It is reset again when the read or write frame is correctly sent, or when set SCR.CDVx.

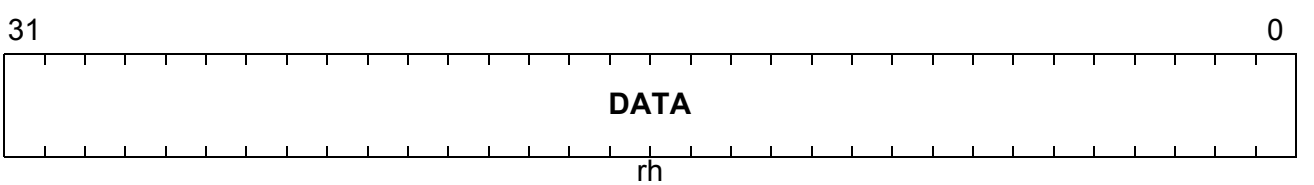
Field	Bits	Type	Description
RPx, x = 0, 1, 2, 3	[23:20]	rh	**Read Pending** Set to one by the MLI transmitter when the TPxAOFR register is written for a read operation. It is reset again when the MLI receiver gets an answer frame for pipe x or when SCR.CDV is set.
PN	[25:24]	rh	**Pipe Number** This bit field will indicate to which pipe number corresponds the base address that has been stored in the TCBAR register, and it is written by the MLI transmitter each time it writes TCBAR with the data from the correspondent TPxBAR register. Its encoding is as follows: 00_B Pipe number 0 01_B Pipe number 1 01_B Pipe number 2 11_B Pipe number 3
0	[15:10] [31:26]	r	**Reserved;** read as 0; should be written with 0.

TP0AOFR
Transmitter Pipe 0 Address Offset Register **Reset Value: 0000 0000$_H$**
TP1AOFR
Transmitter Pipe 1 Address Offset Register **Reset Value: 0000 0000$_H$**
TP2AOFR
Transmitter Pipe 2 Address Offset Register **Reset Value: 0000 0000$_H$**
TP3AOFR
Transmitter Pipe 3 Address Offset Register **Reset Value: 0000 0000$_H$**



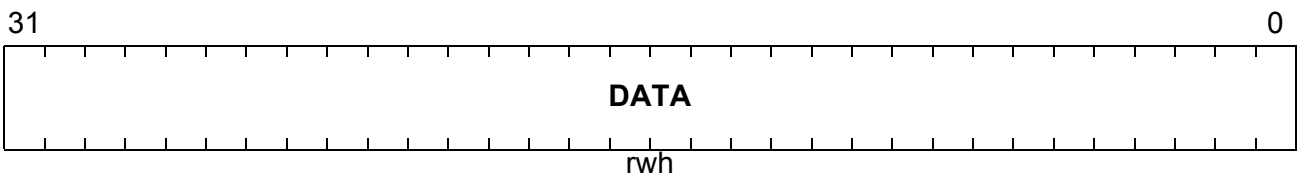
Field	Bits	Type	Description
AOFF	[15:0]	rh	**Address Offset** This address offset together with the base address of the pipe will point to an address position in the transfer window of the other controller in which the controller wants to write or read.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

TP0DATAR
Transmitter Pipe 0 Data Register **Reset Value: 0000 0000$_H$**
TP1DATAR
Transmitter Pipe 1 Data Register **Reset Value: 0000 0000$_H$**
TP2DATAR
Transmitter Pipe 2 Data Register **Reset Value: 0000 0000$_H$**
TP3DATAR
Transmitter Pipe 3 Data Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DATA	[31:0]	rh	**Data** Contains the data that will be sent to the MLI receiver in the other controller through the correspondent pipe. It will be the data to be written and to send it the MLI will use a write frame (optimized or not).

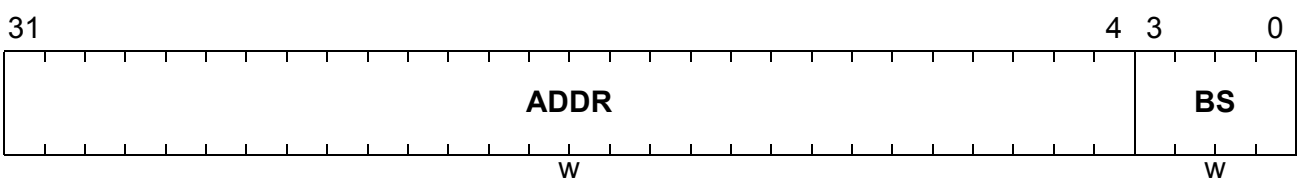
TDRAR
Transmitter Data Read Answer Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DATA	[31:0]	rwh	**Data** Contains the data that proceeds from a read operation and it will be sent to the MLI receiver in the other controller. It will be the data that is sent in answer frame.

When the LSB of TPxBAR register is updated, the 28 MSBs are directly copied in the register TCBAR, and the BS bit field is copied in the correspondent TPxSTATR register.

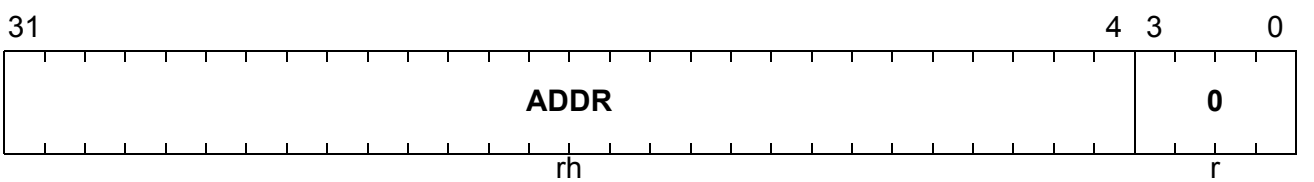
TP0BAR
Transmitter Pipe 0 Base Address Register **Reset Value: 0000 0000$_H$**
TP1BAR
Transmitter Pipe 1 Base Address Register **Reset Value: 0000 0000$_H$**
TP2BAR
Transmitter Pipe 2 Base Address Register **Reset Value: 0000 0000$_H$**
TP3BAR
Transmitter Pipe3 Base Address Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
BS	[3:0]	w	**Buffer Size** It gives the used size of the remote window in the remote controller. The offset width will be coincident with this buffer size field. 0000_B One bit offset 0001_B Two bits offset 0010_B Three bits offset … 1110_B Fifteen bits offset 1111_B Sixteen bits offset
ADDR	[31:4]	w	**Address** This bit field contains the base address 28 MSBs of the correspondent pipe. It is written each time the controller wanted to initialize a pipe in the other controller.

The TCBAR register is updated with the 28 MSBs contained in the latest accessed TPxBAR register. The trigger condition to make the copy is that the LSB of any TPxBAR registers had been written.

TCBAR
Transmitter Copy Base Address Register **Reset Value: 0000 0000_H**

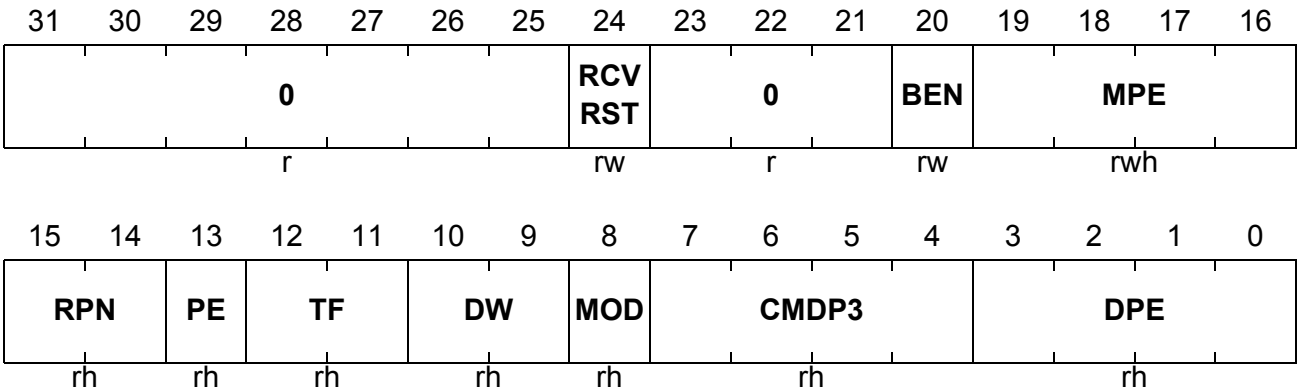


Field	Bits	Type	Description
ADDR	[31:4]	rh	**Address** This bit field contains the base address 28 MSBs of any of the four pipes.
0	[3:0]	r	**Reserved;** read as 0; should be written with 0.

Note: The TRSTATR.BAVx flag indicates if the TCBAR register contains a base address that has been sent or not. The flag is reset again when the copy base address frame has been correctly sent.

27.2.2 MLI Receiver Registers

RCR
Receiver Control Register **Reset Value: 0100 0000$_H$**

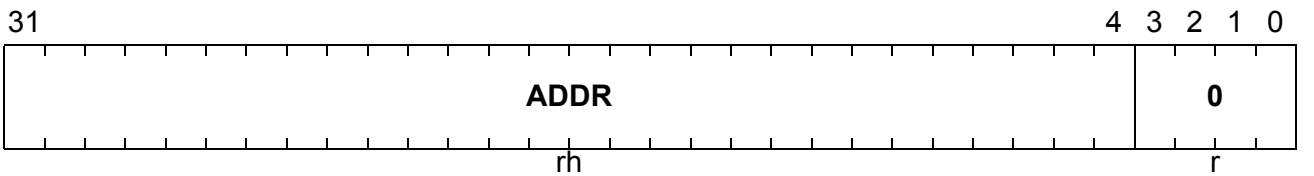


Field	Bits	Type	Description
DPE	[3:0]	rh	**Delay for Parity Error** This bit field is written by the MLI when it receives the proper command to program it. It defines the number of clock periods (from the clock used in the transmission) that as minimum the MLI receiver has to wait to raise again the READY signal when it has detected a parity error. 0000_B Wait zero clock periods 0001_B Wait one clock period 0010_B Wait two clock periods … 1110_B Wait fourteen clock periods 1111_B Wait fifteen clock periods
CMDP3	[7:4]	rh	**Command From Pipe 3** Whenever the MLI receiver gets a command frame trough the pipe 3, it stores its value in this bit field. The command will be interpreted by software.

Field	Bits	Type	Description
MOD	8	rh	**Mode of Operation** This bit field defines the mode of operation of the MLI receiver. The MLI receiver may write this bit field when it receives the proper command from other controller. This bit is also set and reset when its correspondent bits of the clear and set register are set, SCR.CMOD and SCR.SMOD. 0 The automatic read/write handling is disabled. The request is stored in the registers and can be served by the CPU (listening mode). 1 The automatic read/write handling is enabled.
DW	[10:9]	rh	**Data Width** This bit field is updated by the MLI receiver whenever it writes new data in the RDATAR register. Bit field DW is used by the MLI receiver when it delivered the data or by the software whenever it had to fetch data from the MLI receiver RDATAR register. It is updated by read frames, write frames or answer frames. 00_B Data width of 8 bits selected 01_B Data width of 16 bits selected 10_B Data width of 32 bits selected 11_B Reserved
TF	[12:11]	rh	**Type of Frame** Set by the MLI receiver when it writes the RDATAR register. Its value depends on the kind of frame in which the data was received. This bit field will be used by the software in order to know where it must take the data from and how to deliver it. It is updated by copy base address frames, read frames, write frames or answer frames. 00 Copy base address frame 01 Read frame, optimized or not 10 Write frame, optimized or not 11 Answer frame
PE	13	rh	**Parity Error** Set to one when the MLI receiver detects a parity error in the transmission. It is reset again when the MLI receives a transfer without parity error, or when set SCR.CRPE.

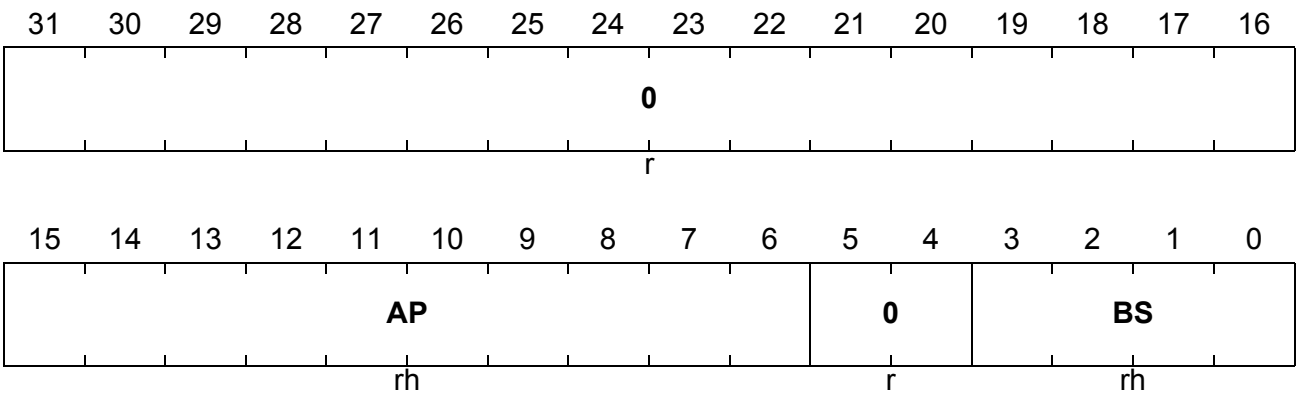
Field	Bits	Type	Description
RPN	[15:14]	rh	**Received Pipe Number** This bit field contains the pipe number that was indicated by the pipe number bit field of the latest received frame. It is updated by copy base address frames, read frames, write frames or answer frames.
MPE	[19:16]	rwh	**Maximum Parity Errors** This bit field indicates after how many parity errors the parity error interrupt will be generated. It is set to a desired value by software and it is decremented automatically by the MLI, each time it detects a parity error. If 0, each parity error will generate an interrupt and MPE stays 0. 0000 Each parity error will generate the interrupt 0001 After 2 parity errors the interrupt is generated 0010 After 3 parity errors the interrupt is generated … 1111 After 16 parity errors the interrupt is generated
BEN	20	rw	**Break Out Enable** If set to one the MLI receiver will produce a pulse in its BREAKOUT line when received the corresponding command frame.
RCVRST	24	rw	**Receiver Reset** This bit forces the receiver to reset in order to be able to change OICR settings without influencing the receiver register. During chip reset, this bit is 1 and in the second clock cycle after the chip reset, the specified reset value is applied. This ensures that no spikes/pulses are received during chip reset (OICR might change). 0 The receiver is not held in reset (operating mode) 1 The receiver is held in reset
0	[23:21], [31:25]	r	**Reserved;** read as 0; should be written with 0.

RP0BAR
Receiver Pipe 0 Base Address Register **Reset Value: 0000 0000$_H$**
RP1BAR
Receiver Pipe 1 Base Address Register **Reset Value: 0000 0000$_H$**
RP2BAR
Receiver Pipe 2 Base Address Register **Reset Value: 0000 0000$_H$**
RP3BAR
Receiver Pipe 3 Base Address Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
ADDR	[31:4]	rh	ADDR indicates the pipe x remote window base address. When a pipe x copy base address frame is received, ADDR [31:4] becomes loaded with the transmitted 28-bit address and bits [3:0] are loaded with 0000_B. For address calculation, ADDR[RPxSTATR.BS-1:0] is used for calculation of the actual read/write address that is stored in register RADDR. ADDR[31:RPxSTATR.BS] remains frozen for this address calculation.
0	[3:0]	r	**Reserved;** read as 0; should be written with 0.

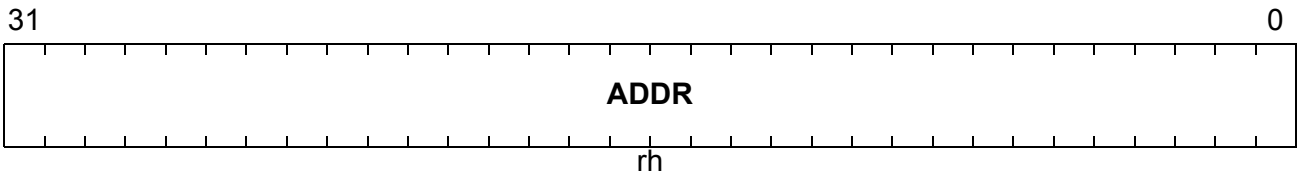
RP0STATR
Receiver Pipe 0 Status Register **Reset Value: 0000 0000$_H$**
RP1STATR
Receiver Pipe 1 Status Register **Reset Value: 0000 0000$_H$**
RP2STATR
Receiver Pipe 2 Status Register **Reset Value: 0000 0000$_H$**
RP3STATR
Receiver Pipe 3 Status Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
BS	[3:0]	rh	**Buffer Size** It is written by the MLI receiver each time it receives a copy base address frame. The offset width of the offset field received in the frames will be coincident with this buffer size field. 0000_B One bit offset 0001_B Two bits offset 0010_B Three bits offset … 1110_B Fifteen bits offset 1111_B Sixteen bits offset
AP	[15:6]	rh	**Address Prediction Factor** It is written by the MLI receiver. It is used to keep track of the address prediction method. It represents a number of ten bits with sign in two's complement.
0	[5:4] [31:16]	r	**Reserved;** read as 0; should be written with 0.

RADRR
Receiver Address Register

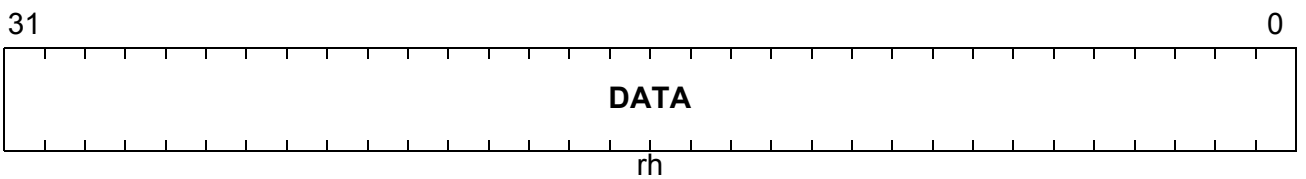
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
ADDR	[31:0]	rh	**Address** It contains the destination absolute address in which the data will be written or from which the data will be read. Its value is totally synchronous to the MLI receiver internal clock.

RDATAR
Receiver Data Register

Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
DATA	[31:0]	rh	**Data** It contains the data that will be written in the controller. It can be data of a write operation or the result from a read operation. Its value is totally synchronous to the MLI receiver internal clock.

27.2.3 MLI Kernel Common Registers

SCR
Set Clear Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCIV3	CCIV2	CCIV1	CCIV0	CNAE	CTPE	CRPE	CAV	0						CBAV	CMOD
w	w	w	w	w	w	w	w	r						w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCV3	CCV2	CCV1	CCV0	CDV3	CDV2	CDV1	CDV0	0			SMOD	SCV3	SCV2	SCV1	SCV0
w	w	w	w	w	w	w	w	r			w	w	w	w	w

Field	Bits	Type	Description
SCVx **x = 0, 1, 2, 3**	[3:0]	w	**Set Command Valid** 0 No effect 1 Bit TRSTATR.CVx is set independently on the value written to SMOD.
SMOD	4	w	**Set MOD Flag** 0 No effect 1 Bit RCR.MOD is set
CDVx **x = 0, 1, 2, 3**	[11:8]	w	**Clear Data Valid 0 Flag** 0 No effect 1 Bits TRSTATR.DVx and TRSTATR.RPx are cleared
CCVx **x = 0, 1, 2, 3**	[15:12]	w	**Clear Command Valid 0 Flag** 0 No effect 1 Bit TRSTATR.CVx is cleared
CMOD	16	w	**Clear MOD Flag** 0 No effect 1 Bit RCR.MOD is cleared independently on the value written to SMOD.
CBAV	17	w	**Clear BAV Flag** 0 No effect 1 Bit TRSTATR.BAV is cleared
CAV	24	w	**Clear AV Flag** 0 No effect 1 Bit TRSTATR.AV is cleared

Field	Bits	Type	Description
CRPE	25	w	**Clear Receiver PE Flag** 0 No effect 1 Bit RCR.PE is cleared
CTPE	26	w	**Clear Transmitter PE Flag** 0 No effect 1 Bit TSTATR.PE is cleared
CNAE	27	w	**Clear NAE Flag** 0 No effect 1 Bit TSTATR.NAE is cleared
CCIVx **x = 0, 1, 2, 3**	[31:28]	w	**Clear Command Interrupt Valid x Flag** 0 No effect 1 Bit TRSTATR.CIVx is cleared
0	[7:5], [23:18]	r	**Reserved;** read as 0; should be written with 0.

Note: The implementation of this register does not involve any flip-flop.

OICR
Output Input Control Register **Reset Value: 1000 8000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDP	**RDS**		**RCE**	**RCP**	**RCS**		**RVP**	**RVS**		**RRPD**	**RRPC**	**RRPB**	**RRPA**	**RRS**	
rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RVE	**TDP**	**TCP**	**TCE**	**TRE**	**TRP**	**TRS**		**TVPD**	**TVPC**	**TVPB**	**TVPA**	**TVED**	**TVEC**	**TVEB**	**TVEA**
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TVEA, **TVEB,** **TVEC,** **TVED**	0, 1, 2, 3	rw	**Transmitter Valid Enable** These bits field enable the transmitter output signals TVALIDx that are driven outside the module. 0 The transmitter output signal TVALIDx is considered as passive. 1 The transmitter output signal TVALIDx reflects the status of the current transmitter kernel signal VALID.

Field	Bits	Type	Description
TVPA, TVPB, TVPC, TVPD	4, 5, 6, 7	rw	**Transmitter Valid Polarity** These bits define the polarity of each of the transmitter output signals TVALIDx. 0 An active TVALIDx line is driving a 1, a passive level is 0 (not inverted). 1 An active TVALIDx line is driving a 0, a passive level is 1 (inverted).
TRS	[9:8]	rw	**Transmitter Ready Selector** This bit field defines the transmitter input line that is used for the transmitter kernel signal READY. 00 TREADYA is selected 01 TREADYB is selected 10 TREADYC is selected 11 TREADYD is selected
TRP	10	rw	**Transmitter Ready Polarity** This bit defines the polarity of the selected transmitter input signal TREADYx. 0 An active TREADYx level is 1, a passive level is 0 (not inverted). 1 An active TREADYx level is 0, a passive level is 1 (inverted).
TRE	11	rw	**Transmitter Ready Enable** This bit enables the input of the transmitter kernel signal READY. 0 The READY signal is considered as passive (internal = 0). 1 The TREADYx line according to the bit fields TRS and TRP is taken into account.
TCE	12	rw	**Transmitter Clock Enable** This bit enables the transmitter kernel signal CLK to be driven outside the module. 0 The transmitter clock signal is considered as passive (internal = 0). 1 The TCLK line reflects the status of the current transmitter kernel signal CLK according to bit TCP.

Field	Bits	Type	Description
TCP	13	rw	**Transmitter Clock Polarity** This bit defines the polarity of the transmitter output signal TCLK. 0 A passive TCLK line is driving a 0 (not inverted). 1 A passive TCLK line is driving a 1 (inverted).
TDP	14	rw	**Transmitter Data Polarity** This bit defines the polarity of the transmitter output signal TCLK. 0 The transmitter kernel signal DATA drives directly the transmitter output line TDATA (not inverted). 1 The transmitter kernel signal DATA is inverted before driving the transmitter output line TDATA.
RVE	15	rw	**Receiver Valid Enable** This bit enables the receiver kernel input signal VALID. 0 The VALID signal is considered as passive (internal = 0). 1 The RVALIDx line according to the bit fields RVS and RVP is taken into account.
RRS	[17:16]	rw	**Receiver Ready Selector** This bit field defines the receiver output signal RREADYx that is driven outside the module by the receiver kernel signal READY. An RREADYx output signal that is not selected is considered as passive and drives a level according to its corresponding bit RRPx. 00 Select RREADYA 01 Select RREADYB 10 Select RREADYC 11 Select RREADYD
RRPA, RRPB, RRPC, RRPD	18, 19, 20, 21	rw	**Receiver Ready Polarity** These bits define the polarity of the receiver output signals RREADYx. 0 An active RREADYx line level is 1, a passive level is 0 (not inverted). 1 An active RREADYx line level is 0, a passive level is 1 (inverted).

Field	Bits	Type	Description
RVS	[23:22]	rw	**Receiver Valid Selector** This bit defines which one of the receiver input signals RVALIDx that is taken as input for the receiver kernel signal VALID. 00 Select RVALIDA 01 Select RVALIDB 10 Select RVALIDC 11 Select RVALIDD
RVP	24	rw	**Receiver Valid Polarity** This bit defines the polarity of the selected receiver input signal RVALIDx. 0 An active RVALIDx level is 1, a passive level is 0 (not inverted). 1 An active RVALIDx level is 0, a passive level is 1 (inverted).
RCS	[26:25]	rw	**Receiver Clock Selector** This bit defines which one of the receiver input signals RCLKx that is taken as input for the receiver kernel signal CLK. 00 Select RCLKA 01 Select RCLKB 10 Select RCLKC 11 Select RCLKD
RCP	27	rw	**Receiver Clock Polarity** This bit defines the polarity of the selected receiver input signal RCLKx. 0 An active RCLKx level is 1, a passive level is 0 (not inverted). 1 An active RCLKx level is 0, a passive level is 1 (inverted).
RCE	28	rw	**Receiver Clock Enable** This bit enables the receiver kernel input signal CLK. 0 The CLK signal is considered as passive (internal = 0). 1 The RCLKx line according to the bit fields RCS and RCP is taken into account.

Field	Bits	Type	Description
RDS	[30:29]	rw	**Receiver Data Selector** This bit defines which one of the receiver input signals RDATAx that is taken as input for the receiver kernel signal DATA. 00 Select RDATAA 01 Select RDATAB 10 Select RDATAC 11 Select RDATAD
RDP	31	rw	**Receiver Data Polarity** This bit defines the polarity of the selected receiver input signal RDATAx. 0 The receiver kernel signal DATA is directly driven by the selected receiver input line RDATAx (not inverted). 1 The receiver kernel signal DATA is driven by the inverted signal from the selected receiver input line RDATAx.

27.2.4 MLI Interrupt Registers

TIER
Transmitter Interrupt Enable Register **Reset Value: 0000 0000$_H$**

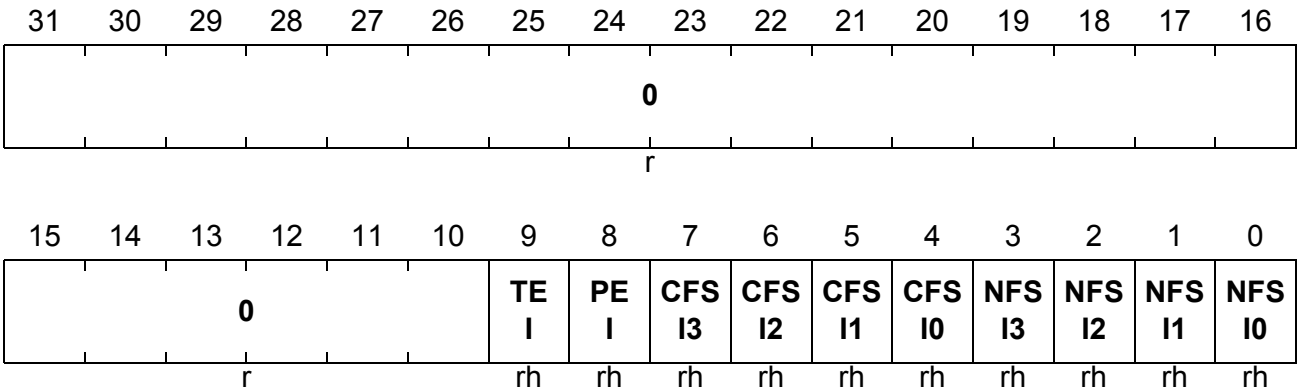
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						TEIR	PEIR	CFSIR3	CFSIR2	CFSIR1	CFSIR0	NFSIR3	NFSIR2	NFSIR1	NFSIR0
r						w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						TEIE	PEIE	CFSIE3	CFSIE2	CFSIE1	CFSIE0	NFSIE3	NFSIE2	NFSIE1	NFSIE0
r						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NFSIEx, x = 0, 1, 2, 3	[3:0]	rw	**Normal Frame Sent in Pipe x Interrupt Enable** If set to one the interrupt NFSIx can be generated.
CFSIEx, x = 0, 1, 2, 3	[7:4]	rw	**Command Frame Sent in Pipe x Interrupt Enable** If set to one the interrupt CFSIx can be generated.
PEIE	8	rw	**Parity Error Interrupt Enable** If set to one the interrupt PEI can be generated.
TEIE	9	rw	**Time-out Error Interrupt Enable** If set to one the interrupt TEI can be generated.
NFSIRx, x = 0, 1, 2, 3	[19:16]	w	**Normal Frame Sent in Pipe x Flag Reset** Writing this bit with 1 resets bit TISR.NFSIx. Writing a 0 has no effect. A read action always delivers 0.
CFSIRx, x = 0, 1, 2, 3	[23:20]	w	**Command Frame Sent in Pipe x Flag Reset** Writing this bit with 1 resets bit TISR.CFSIx. Writing a 0 has no effect. A read action always delivers 0.
PEIR	24	w	**Parity Error or Time-out Error Flag Reset** Writing this bit with 1 resets bit TISR.PEI. Writing a 0 has no effect. A read action always delivers 0.
TEIR	25	w	**Time-out Error Flag Reset** Writing this bit with 1 resets bit TISR.TEI. Writing a 0 has no effect. A read action always delivers 0.
0	[15:10], [31:26]	r	**Reserved;** read as 0; should be written with 0.

TISR
Transmitter Interrupt Status Register

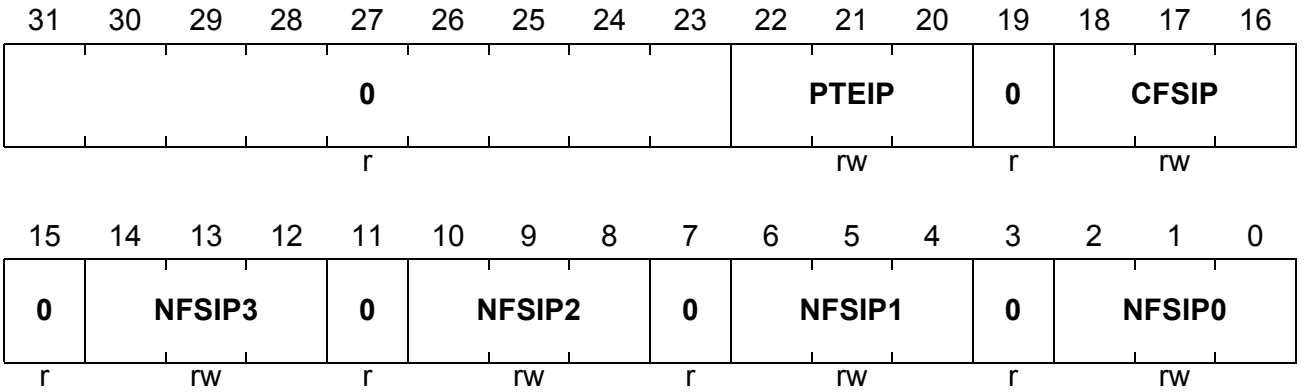
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
NFSIx	[3:0]	rh	**Normal Frame Sent in Pipe x Flag** It is set to one when a normal frame has been sent out correctly on pipe x.
CFSIx	[7:4]	rh	**Command Frame Sent in Pipe x Flag** It is set to one when a command frame has been sent out correctly on pipe x.
PEI	8	rh	**Parity Error Flag** It is set to one when the parity error counter on transmitter side has reached 0.
TEI	9	rh	**Time-out Error Flag** It is set to one when the non acknowledge counter has reached 0 (it is counting down by 1 with each retry due to READY = 1 when VALID becomes 0).
0	[31:10]	r	**Reserved;** read as 0; should be written with 0.

TINPR
Transmitter Interrupt Node Pointer Register

Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
NFSIP0	[2:0]	rw	**Normal Frame Sent in Pipe 0 Interrupt Pointer** Number of the interrupt output reporting the sending of a normal frame through pipe 0 if enabled by NFSIE0 = 1. 000_B MLI interrupt output 0 is selected … 111_B MLI interrupt output 7 is selected
NFSIP1	[6:4]	rw	**Normal Frame Sent in Pipe 1 Interrupt Pointer** Number of the interrupt output reporting the sending of a normal frame through pipe 1 if enabled by NFSIE1 = 1. 000_B MLI interrupt output 0 is selected … 111_B MLI interrupt output 7 is selected
NFSIP2	[10:8]	rw	**Normal Frame Sent in Pipe 2 Interrupt Pointer** Number of the interrupt output reporting the sending of a normal frame through pipe 2 if enabled by NFSIE2 = 1. 000_B MLI interrupt output 0 is selected … 111_B MLI interrupt output 7 is selected

Field	Bits	Type	Description
NFSIP3	[14:12]	rw	**Normal Frame Sent in Pipe 3 Interrupt Pointer** Number of the interrupt output reporting the sending of a normal frame through pipe 3 if enabled by NFSIE3 = 1. 000_B MLI interrupt output 0 is selected … 111_B MLI interrupt output 7 is selected
CFSIP	[18:16]	rw	**Command Frame Sent Interrupt Pointer** Number of the common interrupt output reporting the sending of a command frame for: pipe 0 if enabled by CFSIE0 = 1 or pipe 1 if enabled by CFSIE1 = 1 or pipe 2 if enabled by CFSIE2 = 1 or pipe 3 if enabled by CFSIE3 = 1. 000_B MLI interrupt output 0 is selected … 111_B MLI interrupt output 7 is selected
PTEIP	[22:20]	rw	**Parity or Time-out Interrupt Pointer** Number of the common interrupt output reporting: the parity error if enabled by PEIE = 1 or the time-out interrupt if enabled by TEIE = 1. 000_B MLI interrupt output 0 is selected … 111_B MLI interrupt output 7 is selected
0	3, 7, 11, 15, 19, [31:23]	r	**Reserved;** read as 0; should be written with 0.

RIER
Receiver Interrupt Enable Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						DRAIR	MPEIR	PEIR	ICER	CFRIR3	CFRIR2	CFRIR1	CFRIR0	MEIR	NFRIR
r						w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						DRAIE	MPEIE	PEIE	ICE	CFRIE3	CFRIE2	CFRIE1	CFRIE0	NFRIE	
r						rw	rw	rw	rw	rw	rw	rw	rw	rw	

Field	Bits	Type	Description
NFRIE	[1:0]	rw	**Normal Frame Received Interrupt Enable** This bit field defines if an interrupt is generated when a normal frame is correctly received. 00_B The interrupt generation is disabled 01_B The interrupt is generated each time a normal frame is correctly received 10_B The interrupt is generated each time a normal frame is correctly received that is not automatically handled by the MLI move engine 11_B Reserved
CFRIEx, x = 0, 1, 2, 3	[5:2]	rw	**Command Received through Pipe x Interrupt Enable** This bit defines if an interrupt is generated when a command frame is correctly received through pipe x. 0 The interrupt generation is disabled 1 The interrupt generated is enabled
ICE	6	rw	**Interrupt Command Enable** This bit defines if an interrupt is generated when a command on pipe 0 requests the interrupt generation. The activated interrupt output line is defined by the command. 0 The interrupt generation is disabled 1 The interrupt generated is enabled
PEIE	7	rw	**Parity Error Interrupt Enable** This bit enables the interrupt is generated when a parity error is detected and the parity error counter is 0. 0 The interrupt generation is disabled 1 The interrupt generation is enabled

Field	Bits	Type	Description
MPEIE	8	rw	**Memory Protection Interrupt Enable** This bit enables the interrupt is generated when a memory protection error is detected. 0 The interrupt generation is disabled 1 The interrupt generation is enabled
DRAIE	9	rw	**Discarded Read Answer Interrupt Enable** If set to one the interrupt DRAI can be generated.
NFRIR	16	w	**Normal Frame Received Interrupt Flag Reset** Writing this bit with 1 resets bit RISR.NFRI. Writing a 0 has no effect. A read action always delivers 0.
MEIR	17	w	**MLI Move Engine Interrupt Flag Reset** Writing this bit with 1 resets bit RISR.MEI. Writing a 0 has no effect. A read action always delivers 0.
CFRIRx, x = 0, 1, 2, 3	[21:18]	w	**Command Frame Received through Pipe x Interrupt Flag Reset** Writing this bit with 1 resets bit RISR.CFRIx. Writing a 0 has no effect. A read action always delivers 0.
ICER	22	w	**Interrupt Command Flag Reset** Writing this bit with 1 resets bit RISR.ICE. Writing a 0 has no effect. A read action always delivers 0.
PEIR	23	w	**Parity Error Interrupt Flag Reset** Writing this bit with 1 resets bit RISR.PEI. Writing a 0 has no effect. A read action always delivers 0.
MPEIR	24	w	**Memory Protection Error Interrupt Flag Reset** Writing this bit with 1 resets bit RISR.MPEI. Writing a 0 has no effect. A read action always delivers 0.
DRAIR	25	w	**Discarded Read Answer Interrupt Flag Reset** Writing this bit with 1 resets bit RISR.DRAI. Writing a 0 has no effect. A read action always delivers 0.
0	[15:10] [31:26]	r	**Reserved;** read as 0; should be written with 0.

RISR
Receiver Interrupt Status Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						DRAI	MPEI	PEI	IC	CFR I3	CFR I2	CFR I1	CFR I0	MEI	NFR I
r						rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
NFRI	0	rh	**Normal Frame Received Interrupt Flag** It is set to 1 when received a normal frame.
MEI	1	rh	**MLI Move Engine Interrupt Flag** It is set when the MLI move engine has done the requested transfer.
CFRIx, x = 0, 1, 2, 3	[5:2]	rh	**Command Frame Received through Pipe x Interrupt Flag** It is set to 1 when a command frame has been received correctly on pipe x.
IC	6	rh	**Interrupt Command Flag** It is set to 1 when an interrupt command on pipe 0 requests an interrupt.
PEI	7	rh	**Parity Error Interrupt Flag** It is set to one when the parity error counter on receiver side has reached 0.
MPEI	8	rh	**Memory Protection Error Interrupt Flag** It is set to one when detected a not allowed memory access.
DRAI	9	rh	**Discarded Read Answer Interrupt Flag** It is set to one when the answer to a read command has been discarded, because no read pending flag was set.
0	[31:10]	r	**Reserved;** read as 0; should be written with 0.

RINPR
Receiver Interrupt Node Pointer Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	DRAIP			0	MPPEIP			0	CFRIP			0	NFRIP		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
NFRIP	[2:0]	rw	**Normal Frame Received Interrupt Pointer** Number of the interrupt output reporting the reception of a normal frame if enabled by NFRIE. 000_B MLI interrupt output 0 is selected … 111_B MLI interrupt output 7 is selected
CFRIP	[6:4]	rw	**Command Frame Received Interrupt Pointer** Number of the common interrupt output reporting the reception of a command frame through: pipe 0 if enabled by CFRIE0 = 1 or pipe 1 if enabled by CFRIE1 = 1 or pipe 2 if enabled by CFRIE2 = 1 or pipe 3 if enabled by CFRIE3 = 1. 000_B MLI interrupt output 0 is selected … 111_B MLI interrupt output 7 is selected
MPPEIP	[10:8]	rw	**Memory Protection or Parity Error Interrupt Pointer** Number of the interrupt output reporting a parity error or a memory protection error on receiver side if enabled by MPEIE = 1 or PEIE = 1. 000_B MLI interrupt output 0 is selected … 111_B MLI interrupt output 7 is selected

Field	Bits	Type	Description
DRAIP	[14:12]	rw	**Discarded Read Answer Interrupt Pointer** Number of the interrupt output reporting that a read answer has been discarded if enabled by DRAIE = 1. 000_B MLI interrupt output 0 is selected … 111_B MLI interrupt output 7 is selected
0	3, 7, 11, [31:15]	r	**Reserved;** read as 0; should be written with 0.

The GINTR can activate the interrupt output lines of the MLI module. The implementation of this register does not involve any flip-flop.

GINTR
Global Interrupt Set Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								SI MLI7	SI MLI6	SI MLI5	SI MLI4	SI MLI3	SI MLI2	SI MLI1	SI MLI0
r								w	w	w	w	w	w	w	w

Field	Bits	Type	Description
SIMLIx (x = 0-7)	x	w	**Set MLI Interrupt Output Line x** 0 No action 1 The MLI interrupt output line x will be activated
0	[31:8]	r	**Reserved;** read as 0; should be written with 0.

27.2.5 Memory Protection Registers

The AER register enables write and read operations in the corresponding address ranges (x = 0 to 31). Each address range can be individually enabled.

AER
Access Enable Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AEN 31	**AEN 30**	**AEN 29**	**AEN 28**	**AEN 27**	**AEN 26**	**AEN 25**	**AEN 24**	**AEN 23**	**AEN 22**	**AEN 21**	**AEN 20**	**AEN 19**	**AEN 18**	**AEN 17**	**AEN 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AEN 15	**AEN 14**	**AEN 13**	**AEN 12**	**AEN 11**	**AEN 10**	**AEN 9**	**AEN 8**	**AEN 7**	**AEN 6**	**AEN 5**	**AEN 4**	**AEN 3**	**AEN 2**	**AEN 1**	**AEN 0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
AENx (x = 0-31)	x	rw	**Access Enable** This bit enables the read and write capability of the MLI from the address range x. (x = 31 - 0) 0 The MLI read and write action to this address range is disabled 1 The MLI read and write action to this address range is enabled

Note: The address ranges related to these bits are described in the implementation section for this module.These registers are ENDINIT-protected.

The ARR register selects the address range (only for internal memories, not for modules) that can be accessed by the MLI if the corresponding address range is enabled.

ARR
Access Range Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SIZE3			SLICE3					SIZE2			SLICE2				
rw			rw					rw			rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIZE1			SLICE1					SIZE0			SLICE0				
rw			rw					rw			rw				

Field	Bits	Type	Description
SLICE0, SLICE1, SLICE2, SLICE3	[4:0], [12:8], [20:16], [28:24]	rw	**Address Slice x (x = 0, 1, 2, 3)** This bit field defines which part of the memory address range x can be accessed by the MLI, if enabled.
SIZE0, SIZE1, SIZE2, SIZE3	[7:5], [15:13], [23:21], [31:29]	rw	**Address Size x (x = 0, 1, 2, 3)** This bit field defines which size of the memory address range x can be accessed by the MLI, if enabled.

Note: The address ranges related to these bits are described in the implementation section for this module.These registers are ENDINIT-protected.

27.3 MLI0/MLI1 Module Implementation

This section describes the MLI0/MLI1 Module interfaces with the clock control, port connections, interrupt control, and address decoding.

27.3.1 Interfaces of the MLI Modules

27.3.1.1 Port Connections of MLI0

Figure 27-47 shows the TC1130 specific implementation details and interconnections of the MLI0 module. It supplied by a separate clock control, interrupt control, address decoding, and port control logic.

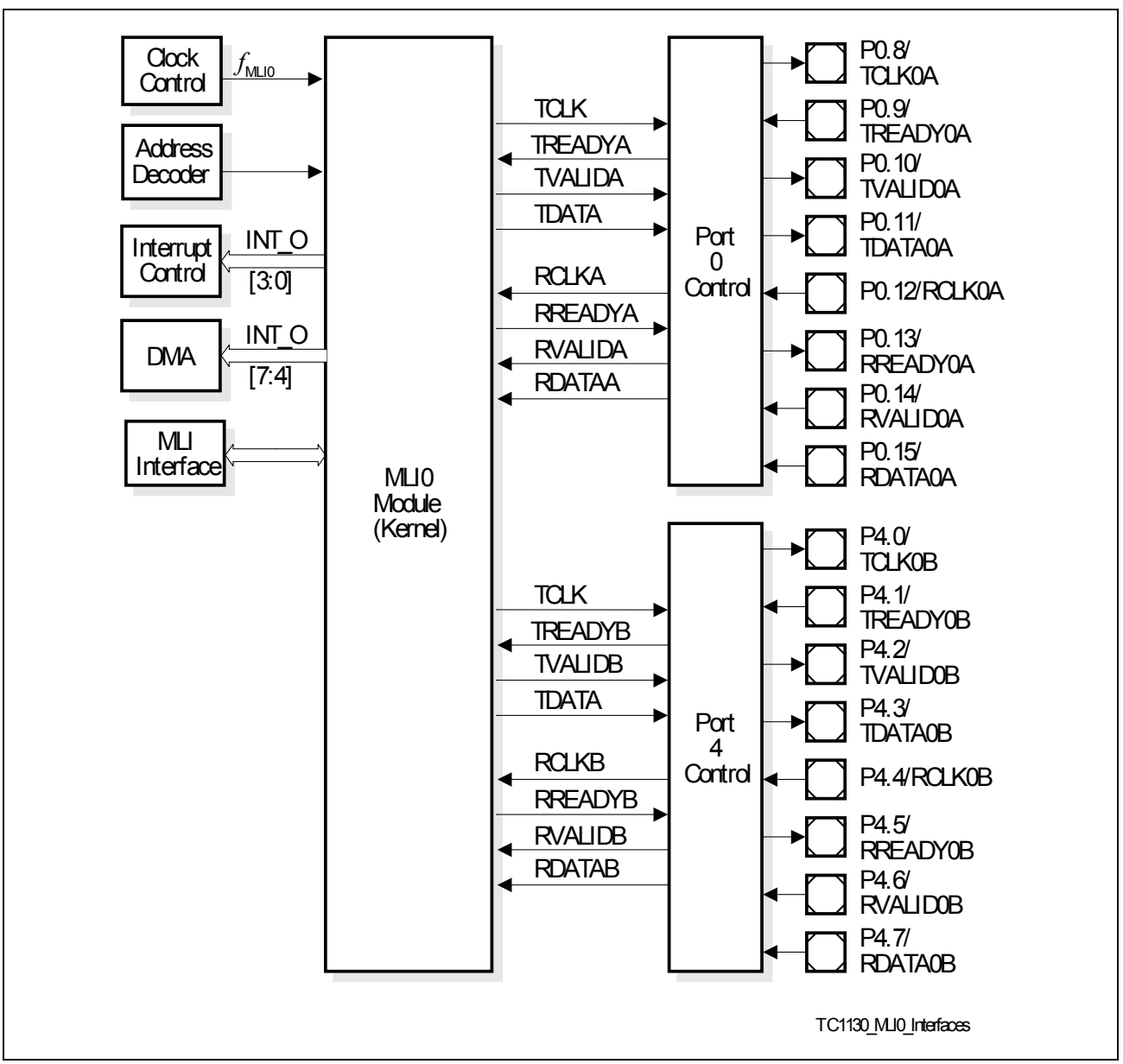


Figure 27-47 MLI0 Module Implementation and Interconnections

Table 27-27 MLI0 Signal Connections

MLI0 Signals	Input/ Output	Connected to
TCLK	O	Port P0.8 (named TCLK0A) Port P4.0 (named TCLK0B) RCLKD of MLI0 (loop back mode)
TREADYA	I	Port P0.9 (named TREADY0A)
TREADYB	I	Port P4.1 (named TREADY0B)
TREADYC	I	0
TREADYD	I	RREADYD of MLI0 (loop back mode)
TVALIDA	O	Port P0.10 (named TVALID0A)
TVALIDB	O	Port P4.2 (named TVALID0B)
TVALIDC	O	Left open
TVALIDD	O	RVALIDD of MLI0 (loop back mode)
TDATA	O	Port P0.11 (named TDATA0A) Port P4.3 (named TDATA0B) RDATAD of MLI0 (loop back mode)
RCLKA	I	Port P0.12 (named RCLK0A)
RCLKB	I	Port P4.4 (named RCLK0B)
RCLKC	I	0
RCLKD	I	TCLK of MLI0 (loop back mode)
RREADYA	O	Port P0.13 (named RREADY0A)
RREADYB	O	Port P4.5 (named RREADY0B)
RREADYC	O	Left open
RREADYD	O	TREADYD of MLI0 (loop back mode)
RVALIDA	I	Port P0.14 (named RVALID0A)
RVALIDB	I	Port P4.6 (named RVALID0B)
RVALIDC	I	0
RVALIDD	I	TVALIDD of MLI0 (loop back mode)
RDATAA	I	Port P0.15 (named RDATA0A)
RDATAB	I	Port P4.7 (named RDATA0B)
RDATAC	I	0
RDATAD	I	TDATA of MLI0 (loop back mode)

27.3.1.2 Port Connections of MLI1

Figure 27-48 shows the TC1130 specific implementation details and interconnections of the MLI1 module. It supplied by a separate clock control, interrupt control, address decoding, and port control logic.

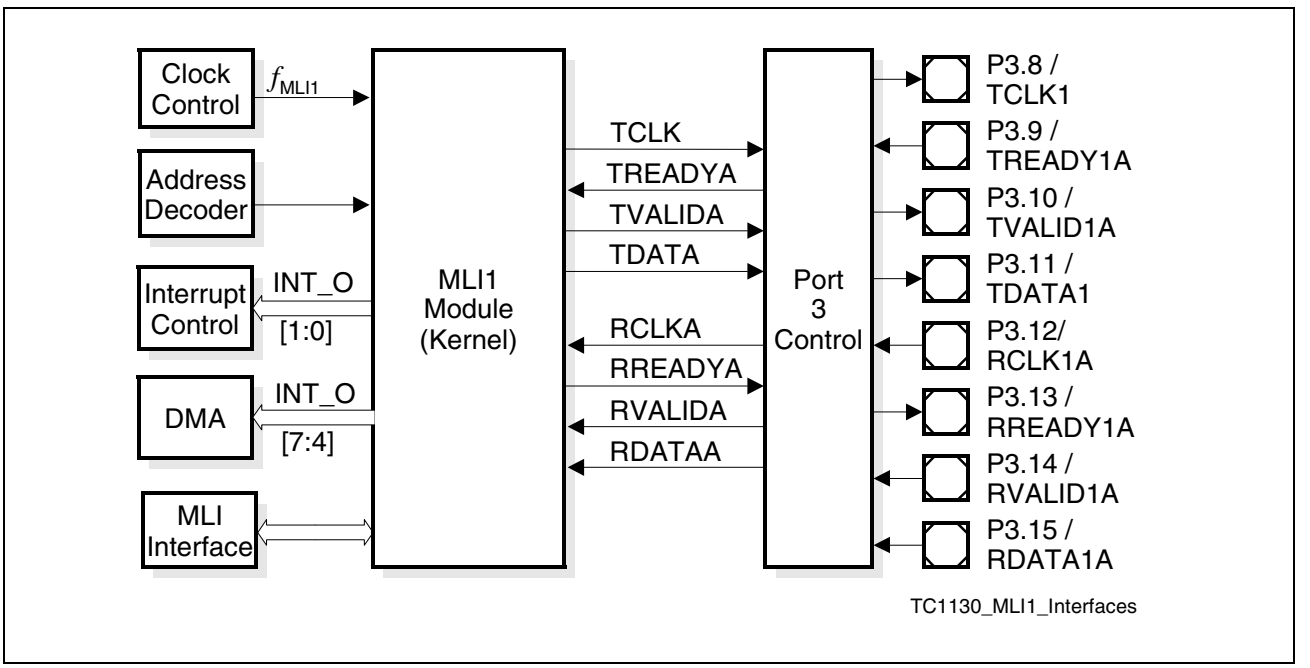


Figure 27-48 MLI1 Module Implementation and Interconnections

Table 27-28 MLI1 Signal Connections

MLI1 Signals	Input/ Output	Connected to
TCLK	O	Port P3.8 (named TCLK1) RCLKD of MLI1 (loop back mode)
TREADYA	I	Port P3.9 (named TREADY1A)
TREADYB	I	0
TREADYC	I	0
TREADYD	I	RREADYD of MLI1 (loop back mode)
TVALIDA	O	Port P3.10 (named TVALID1A)
TVALIDB	O	Left open
TVALIDC	O	Left open
TVALIDD	O	RVALIDD of MLI1 (loop back mode)
TDATA	O	Port P3.11 (named TDATA1) RDATAD of MLI1 (loop back mode)

Table 27-28 MLI1 Signal Connections (cont'd)

MLI1 Signals	Input/ Output	Connected to
RCLKA	I	Port P3.12 (named RCLK1A)
RCLKB	I	0
RCLKC	I	0
RCLKD	I	TCLK of MLI1 (loop back mode)
RREADYA	O	Port P3.13 (named RREADY1A)
RREADYB	O	Left open
RREADYC	O	Left open
RREADYD	O	TREADYD of MLI1 (loop back mode)
RVALIDA	I	Port P3.14 (named RVALID1A)
RVALIDB	I	0
RVALIDC	I	0
RVALIDD	I	TVALIDD of MLI1 (loop back mode)
RDATAA	I	Port P3.15 (named RDATA1A)
RDATAB	I	0
RDATAC	I	0
RDATAD	I	TDATA of MLI1 (loop back mode)

Note: The naming of the port signals has been done according to the following scheme:
SIGNALNAMExn with:
x = module number (0, 1)
n = line of the corresponding vector (A, B, C, D) e.g.
TCLK0 is the transmit clock of MLI0,
TCLK1 is the transmit clock of MLI1,
TREADY0A is the transmit ready input A of MLI0,
TREADY1A is the transmit ready input A of MLI1, etc.

27.3.2 Access Protection

The table below shows the address ranges covered by the access protection (for read and write) of the MLI module. The access enable bits are located in the register AER. The bit at the bit position x in this register is related to the address range x in Table 27-29. Some bits can cover several address ranges (cluster of modules). In the case that a read or a write access has been requested with the corresponding enable bit = 0, an error interrupt is generated.

Table 27-29 DMA Access Protection Address Ranges

Range Number	Related Enable Bits	Covered Address Range	Corresponding to Modules
x = 0	ME0AENR.AEN0	F000 0000$_H$ to F000 00FF$_H$ F010 C200$_H$ to F010 C2FF$_H$	SCU, incl. WDT, MEMCHK
x = 1	ME0AENR.AEN1	F000 0100$_H$ to F000 01FF$_H$	SBCU
x = 2	ME0AENR.AEN2	F000 0200$_H$ to F000 02FF$_H$	STM
x = 3	ME0AENR.AEN3	F000 0300$_H$ to F000 03FF$_H$	OCDS
x = 4	ME0AENR.AEN4	F000 0600$_H$ to F000 06FF$_H$	GPTU
x = 5	ME0AENR.AEN5	F000 0C00$_H$ to F000 10FF$_H$	P0, P1, P2, P3, P4
x = 6	ME0AENR.AEN6	F000 3C00$_H$ to F000 3EFF$_H$	DMA
x = 7	ME0AENR.AEN7	F000 4000$_H$ to F000 5FFF$_H$	MultiCAN
x = 8	ME0AENR.AEN8	F010 0600$_H$ to F010 06FF$_H$	IIC
x = 9	ME0AENR.AEN9	F000 2000$_H$ to F000 20FF$_H$	CCU0
x = 10	ME0AENR.AEN10	F000 2100$_H$ to F000 21FF$_H$	CCU1
x = 11	ME0AENR.AEN11	–	–
x = 12	ME0AENR.AEN12	–	–
x = 13	ME0AENR.AEN13	F010 0100$_H$ to F010 01FF$_H$	SSC0
x = 14	ME0AENR.AEN14	F010 0200$_H$ to F010 02FF$_H$	SSC1
x = 15	ME0AENR.AEN15	F010 0300$_H$ to F010 03FF$_H$	ASC0
x = 16	ME0AENR.AEN16	F010 0200$_H$ to F010 02FF$_H$	ASC1
x = 17	ME0AENR.AEN17	F010 0500$_H$ to F010 05FF$_H$	ASC2
x = 18	ME0AENR.AEN18	–	–
x = 19	ME0AENR.AEN19	F010 C000$_H$ to F010 C0FF$_H$ F01E 0000$_H$ to F01E 7FFF$_H$ F020 0000$_H$ to F023 FFFF$_H$	MLI0 Module, MLI0 Small TWs, MLI0 Large TWs

Table 27-29 DMA Access Protection Address Ranges (cont'd)

Range Number	Related Enable Bits	Covered Address Range	Corresponding to Modules
x = 20	ME0AENR.AEN20	F010 $C100_H$ to F010 $C1FF_H$ F01E 8000_H to F01E $FFFF_H$ F024 0000_H to F027 $FFFF_H$	MLI1, MLI1 Small TWs, MLI1 Large TWs
x = 21	ME0AENR.AEN21	F7E0 $FF00_H$ to F7E0 $FFFF_H$ F7E1 0000_H to F7E1 $FFFF_H$ F800 0400_H to F87F $FFFF_H$	CPS, MMU, CPU, SFRs & GPRs, DMU, DMI, PMI, LBCU, LFI
x = 22	ME0AENR.AEN22	F800 000_H to F800 $03FF_H$	EBU
x = 23	ME0AENR.AEN23	–	–
x = 24	ME0AENR.AEN24	8000 0000_H to 8FFF $FFFF_H$ A000 0000_H to AFFF $FFFF_H$	Ext. EBU Space
x = 25	ME0AENR.AEN25	–	–
x = 26	ME0AENR.AEN26	–	–
x = 27	ME0AENR.AEN27	D800 0000_H to DEFF $FFFF_H$ E000 0000_H to E7FF $FFFF_H$	External Peripherals & External Emulator
x = 28	ME0AENR.AEN28	DFFF $C000_H$ to $DFFFFFFF_H$	Boot ROM
x = 29	ME0AENR.AEN29	E800 0000_H to E83F $FFFF_H$	DMU Image (E80x translated to C00x)
x = 30	ME0AENR.AEN30	E840 0000_H to E84F $FFFF_H$	DMI Image (E84x translated to D00x)
x = 31	ME0AENR.AEN31	E850 0000_H to E85F $FFFF_H$	PMI Image (E85x translated to D40x)

The internal memory blocks are protected by an address range verification in addition to the access enable bits. The address range verification is based on the bit fields SIZEx and SLICEx, which are located in the registers ARR. Only accesses targeting the enabled (by the corresponding AEN bit) and selected memory area (value given by SIZEx, SLICEx) can be handled automatically. Automatic accesses to other locations are not supported.

In order to keep a similar structure for the access protection structure, the assignment is kept identical, although not always the complete range will be available (depending on the memory implementation of different devices).

The address ranges described by SLICEx and SIZEx are defined as follows:

- SIZE0, SLICE0:
  8 Kbytes address range from F010 A000$_H$ to F010 BFFF$_H$,
  this range is unused in TC1130
- SIZE1, SLICE1:
  64 Kbytes address range from E800 0000$_H$ to E800 FFFF$_H$,
  with address translation from E800 to C000, covering the internal SRAM area
- SIZE2, SLICE2:
  64 Kbytes address range from E840 0000$_H$ to E840 FFFF$_H$,
  with address translation from E800 to D000, covering the DMI RAM area
- SIZE3, SLIZE3:
  address range reserved for future extensions,
  this range is unused in TC1130

For the internal SRAM (with address translation from E800 to C000 in the LFI):

Table 27-30 SRAM Read/Write Address Range Verification

Bit Field SIZE1	Size of the Available Address Slice	Bit Field SLICE1	Available Address Range
000	512 Bytes	00000 00001 … 11111	E800 0000$_H$ to E800 01FF$_H$ E800 0200$_H$ to E800 03FF$_H$ E800 3E00$_H$ to E800 3FFF$_H$
001	1 Kbyte	00000 00001 … 11111	E800 0000$_H$ to E800 03FF$_H$ E800 0400$_H$ to E800 07FF$_H$ E800 7C00$_H$ to E800 7FFF$_H$
010	2 Kbytes	00000 00001 … 11111	E800 0000$_H$ to E800 07FF$_H$ E800 0800$_H$ to E800 0FFF$_H$ E800 F800$_H$ to E800 FFFF$_H$
011	4 Kbytes	X0000 X0001 … X1111	E800 0000$_H$ to E800 0FFF$_H$ E800 1000$_H$ to E800 1FFF$_H$ E800 F000$_H$ to E800 FFFF$_H$
100	8 Kbytes	XX000 XX001 … XX111	E800 0000$_H$ to E800 1FFF$_H$ E800 2000$_H$ to E800 3FFF$_H$ E800 E000$_H$ to E800 FFFF$_H$

Table 27-30 SRAM Read/Write Address Range Verification (cont'd)

Bit Field SIZE1	Size of the Available Address Slice	Bit Field SLICE1	Available Address Range
101	16 Kbytes	XXX00 XXX01 XXX10 XXX11	E800 0000$_H$ to E800 3FFF$_H$ E800 4000$_H$ to E800 7FFF$_H$ E800 8000$_H$ to E800 BFFF$_H$ E800 C000$_H$ to E800 FFFF$_H$
110	32 Kbytes	XXXX0 XXXX1	E800 0000$_H$ to E800 7FFF$_H$ E800 8000$_H$ to E800 FFFF$_H$
111	64 Kbytes	XXXXX	E800 0000$_H$ to E800 FFFF$_H$

For the internal DMI SPRAM (with address translation from E840 to D000 in the LFI):

Table 27-31 DMI RAM Read/Write Address Range Verification

Bit Field SIZE2	Size of the Available Address Slice	Bit Field SLICE2	Available Address Range
000	512 Bytes	00000 00001 … 11111	E840 0000$_H$ to E840 01FF$_H$ E840 0200$_H$ to E840 03FF$_H$ E840 3E00$_H$ to E840 3FFF$_H$
001	1 Kbyte	00000 00001 … 11111	E840 0000$_H$ to E840 03FF$_H$ E840 0400$_H$ to E840 07FF$_H$ E840 7C00$_H$ to E840 7FFF$_H$
010	2 Kbytes	00000 00001 … 01111 1XXXX	E840 0000$_H$ to E840 07FF$_H$ E840 0800$_H$ to E840 0FFF$_H$ E840 7C00$_H$ to E840 7FFF$_H$ Not Valid
011	4 Kbytes	X0000 X0001 … X0111 X1XXX	E840 0000$_H$ to E840 0FFF$_H$ E840 1000$_H$ to E840 1FFF$_H$ E840 7000$_H$ to E840 7FFF$_H$ Not Valid

Table 27-31 DMI RAM Read/Write Address Range Verification (cont'd)

Bit Field SIZE2	Size of the Available Address Slice	Bit Field SLICE2	Available Address Range
100	8 Kbytes	XX000 XX001 XX010 XX011 XX1XX	E840 0000_H to E840 $1FFF_H$ E840 2000_H to E840 $3FFF_H$ E840 4000_H to E840 $5FFF_H$ E840 6000_H to E840 $7FFF_H$ Not Valid
101	16 Kbytes	XXX00 XXX01 XXX1X	E840 0000_H to E840 $3FFF_H$ E840 4000_H to E840 $7FFF_H$ Not Valid
110	32 Kbytes	XXXX0 XXXX1	E840 0000_H to E840 $7FFF_H$ Not Valid
111	64 Kbytes	XXXXX	E840 0000_H to E840 $7FFF_H$ E840 8000_H to E840 $FFFF_H$ Not Valid

27.3.3 MLI0/MLI1 Module Related External Registers

Figure 27-49 summarizes the module related external registers which are required for MLI0/MLI1 programming (see also **Figure 27-46** for the module kernel specific registers).

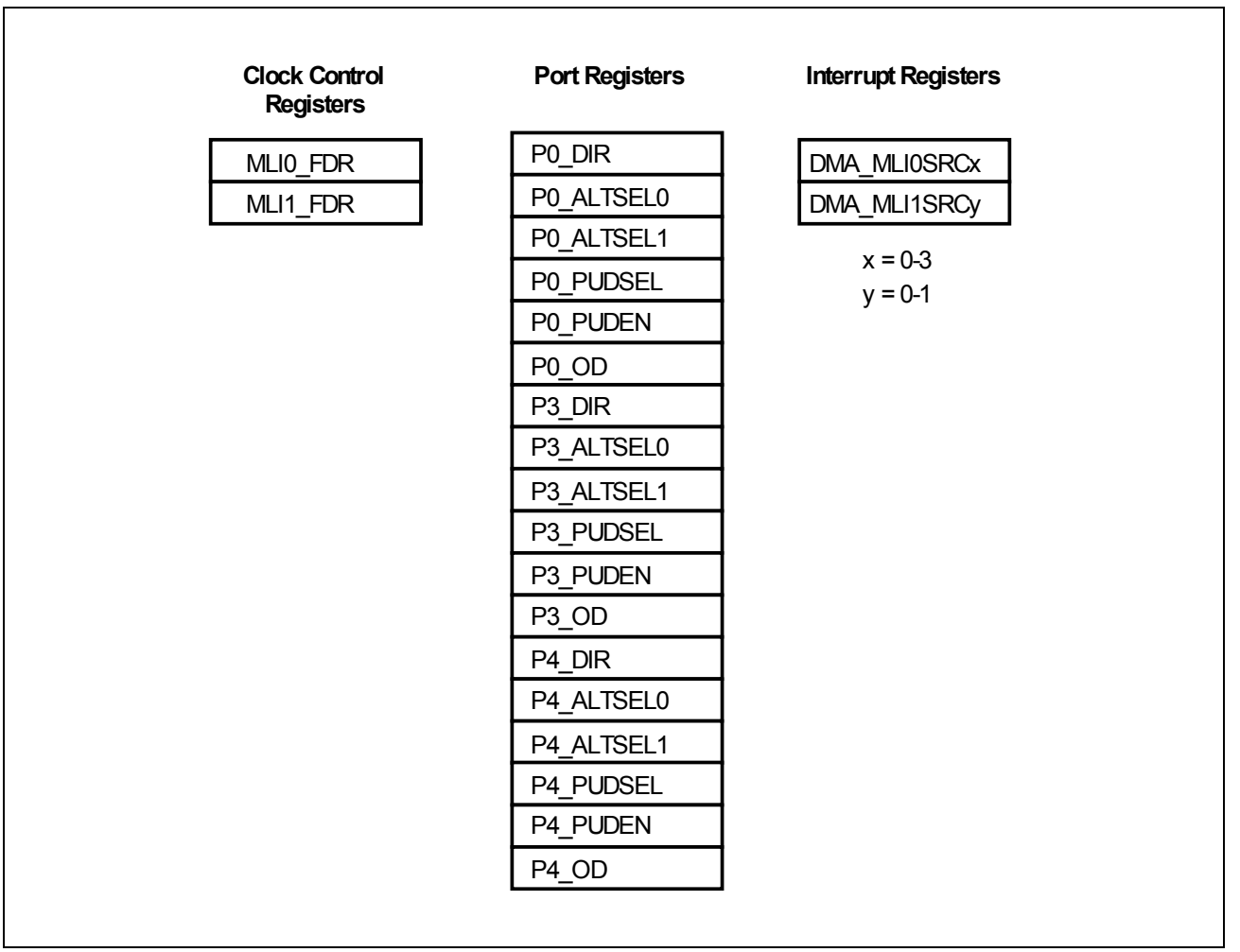


Figure 27-49 MLI0/MLI1 Implementation Specific Special Function Registers

Note: In the current version the four lines trigger_command and their related registers are not implemented. Further, the interrupt registers are described in the chapter "Direct Memory Access Controller" of the TC1130 System Units User's Manual.

27.3.3.1 DMA Requests

Each MLI interface provides DMA request output lines to the DMA controller. These are the interrupt output lines INT_O[7:4] for MLI0 and the lines INT_O[7:4] for MLI1.

27.3.3.2 Interrupt Registers

Each MLI module has eight interrupt request outputs. For MLI0, the lower four lines (INT_O[3:0]) are connected to service request nodes, whereas for MLI1, only the lower two lines (INT_O[1:0]) are connected to service request nodes. These registers are described in the chapter “Direct Memory Access Controller” of the TC1130 System Units User’s Manual.

27.3.3.3 Fractional Divider Registers

The fractional divider register allows the programmer to control the clock rate and period of the 50% duty cycle shift clock f_{MLI}. The period of f_{MLI} can be either 1/STEP or a fraction of STEP/1024 (for any value of STEP from 0 to 1023) of clock f_{MLI}. Each MLI has its own fractional divider.

MLI0_FDR
MLI0 Fractional Divider Register **Reset Value: 03FF 43FF$_H$**
MLI1_FDR
MLI1 Fractional Divider Register **Reset Value: 03FF 43FF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIS CLK	EN HW	SUS REQ	SUS ACK	0		RESULT									
rwh	rw	rh	rh	r		rh									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DM		SC		SM	0	STEP									
rw		rw		rw	r	rw									

Field	Bits	Type	Description
STEP	[9:0]	rw	**Step Value** Reload or addition value for RESULT.
SM	11	rw	**Suspend Mode** 0 Granted suspend mode 1 Immediate suspend mode
SC	[13:12]	rw	**Divider Mode** This bit field selects normal divider mode or fractional divider mode.
DM	[15:14]	rw	**Suspend Control** This bit field defines the behavior of the fractional divider in suspend mode.

Field	Bits	Type	Description
RESULT	[25:16]	rh	**Result Value** Bit fields for the addition result.
SUSACK	28	rh	**Suspend Mode Acknowledge** Indicates state of SPNDACK signal.
SUSREQ	29	rh	**Suspend Mode Request** Indicates state of SPND signal.
ENHW	30	rw	**Enable Hardware Clock Control** Controls operation of ECEN input and DISCLK bit.
DISCLK	31	rwh	**Disable Clock** Hardware controlled disable for f_{OUT} signal.
0	10, [27:26]	r	**Reserved;** read as 0; should be written with 0.

27.3.3.4 Port Control

The interconnections between the MLI modules and the port I/O lines are controlled in the port logics. The following port control operation selections must be executed (additionally to the PISEL programming):

- Input/output direction selection (DIR registers)
- Alternate function selection (ALTSEL0 and ALTSEL1 registers)
- Input/Output driver characteristic control (PUDSEL, PUDEN and OD registers)

The port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as pull-up/down devices, port direction (input/output), open-drain, and alternate output selections. The I/O lines for the MLI modules are controlled by the port input/output control registers of Port 0, Port 3 and Port 4.

Table 27-32 shows how bits and bit fields must be programmed for the required I/O functionality of the MLI I/O lines.

Table 27-32 MLI0 and MLI1 I/O Line Selection and Setup

Module	Port Lines	Input/Output Control Register Bits	I/O
MLI0	P0.8/TCLK0A	P0_DIR.P8 = 1_B	Output
		P0_ALTSEL0.P8 = 0_B	
		P0_ALTSEL1.P8 = 1_B	
	P0.9/TREADY0A	P0_DIR.P9 = 0_B	Input
	P0.10/TVALID0A	P0_DIR.P10 = 1_B	Output
		P0_ALTSEL0.P10 = 0_B	
		P0_ALTSEL1.P10 = 1_B	
	P0.11/TDATA0A	P0_DIR.P11 = 1_B	Output
		P0_ALTSEL0.P11 = 0_B	
		P0_ALTSEL1.P11 = 1_B	
	P0.12/RCLK0A	P0_DIR.P12 = 0_B	Input
	P0.13/RREADY0A	P0_DIR.P13 = 1_B	Output
		P0_ALTSEL0.P13 = 0_B	
		P0_ALTSEL1.P13 = 1_B	
	P0.14/RVALID0A	P0_DIR.P14 = 0_B	Input
	P0.15/RDATA0A	P0_DIR.P15 = 0_B	Input
	P4.0/TCLK0B	P4_DIR.P0 = 1_B	Output
		P4_ALTSEL0.P0 = 0_B	
		P4_ALTSEL1.P0 = 1_B	
	P4.1/TREADY0B	P4_DIR.P1 = 0_B	Input
	P4.2/TVALID0B	P4_DIR.P2 = 1_B	Output
		P4_ALTSEL0.P2 = 0_B	
		P4_ALTSEL1.P2 = 1_B	
	P4.3/TDATA0B	P4_DIR.P3 = 1_B	Output
		P4_ALTSEL0.P3 = 0_B	
		P4_ALTSEL1.P3 = 1_B	
	P4.4/RCLK0B	P4_DIR.P4 = 0_B	Input

Table 27-32 MLI0 and MLI1 I/O Line Selection and Setup (cont'd)

Module	Port Lines	Input/Output Control Register Bits	I/O
	P4.5/RREADY0B	P4_DIR.P5 = 1_B	Output
		P4_ALTSEL0.P5 = 0_B	
		P4_ALTSEL1.P5 = 1_B	
	P4.6/RVALID0B	P4_DIR.P6 = 0_B	Input
	P4.7/RDATA0B	P4_DIR.P7 = 0_B	Input
MLI1	P3.8/TCLK1	P3_DIR.P8 = 1_B	Output
		P3_ALTSEL0.P8 = 0_B	
		P3_ALTSEL1.P8 = 1_B	
	P3.9/TREADY1	P3_DIR.P9 = 0_B	Input
	P3.10/TVALID1	P3_DIR.P10 = 1_B	Output
		P3_ALTSEL0.P10 = 0_B	
		P3_ALTSEL1.P10 = 1_B	
	P3.11/TDATA1	P3_DIR.P11 = 1_B	Output
		P3_ALTSEL0.P11 = 0_B	
		P3_ALTSEL1.P11 = 1_B	
	P3.12/RCLK1	P3_DIR.P12 = 0_B	Input
	P3.13/RREADY1	P3_DIR.P13 = 1_B	Output
		P3_ALTSEL0.P13 = 0_B	
		P3_ALTSEL1.P13 = 1_B	
	P3.14/RVALID1	P3_DIR.P14 = 0_B	Input
	P3.15/RDATA1	P3_DIR.P15 = 0_B	Input

P0_DIR
Port 0 Direction Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 8-15)**	n	rw	**Port 0 Pin 8 - 15 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for MLI I/O port control.

P3_DIR
Port 3 Direction Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 8-15)**	n	rw	**Port 0 Pin 8 - 15 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for MLI I/O port control.

P4_DIR
Port 4 Direction Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								P7	P6	P5	P4	P3	P2	P1	P0
r								rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-7)**	n	rw	**Port 0 Pin 0 - 7 Direction Control** 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:8]	r	**Reserved;** read as 0; should be written with 0.

P0_ALTSELn (n = 0, 1)
Port 0 Alternate Select Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 27-33 Function of the Bits P0_ALTSEL0.Pn and P0_ALTSEL1.Pn (n = 8-15)[1)]

P0_ALTSEL0.Pn	P0_ALTSEL1.Pn	Function
0	1	Alternate Select 2

1) Shaded bits and bit field are don't care for MLI I/O port control.

P3_ALTSELn (n = 0, 1)
Port 3 Alternate Select Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 27-34 Function of the Bits P3_ALTSEL0.Pn and P3_ALTSEL1.Pn (n = 8-15)[1)]

P3_ALTSEL0.Pn	P3_ALTSEL1.Pn	Function
0	1	Alternate Select 2

1) Shaded bits and bit field are don't care for MLI I/O port control.

P4_ALTSELn (n = 0, 1)
Port 4 Alternate Select Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								P7	P6	P5	P4	P3	P2	P1	P0
r								rw	rw	rw	rw	rw	rw	rw	rw

Table 27-35 Function of the Bits P4_ALTSEL0.Pn and P4_ALTSEL1.Pn (n = 0-7)

P4_ALTSEL0.Pn	P4_ALTSEL1.Pn	Function
0	1	Alternate Select 2

P0_OD
Port 0 Open Drain Control Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 8-15)**	n	rw	**Port 0 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for MLI I/O port control.

P3_OD
Port 3 Open Drain Control Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 8-15)**	n	rw	**Port 3 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for MLI I/O port control.

P4_OD
Port 4 Open Drain Control Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								P7	P6	P5	P4	P3	P2	P1	P0
r								rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-7)**	n	rw	**Port 4 Pin n Open Drain Mode** 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:8]	r	**Reserved;** read as 0; should be written with 0.

P0_PUDSEL
Port 0 Pull-Up/Pull-Down Select Register

Reset Value: 0000 FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 8-15)**	n	rw	**Pull-Up/Pull-Down Select Port 0 Bit n**[1)] 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for MLI I/O port control.

P3_PUDSEL
Port 3 Pull-Up/Pull-Down Select Register

Reset Value: 0000 FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 8-15)**	n	rw	**Pull-Up/Pull-Down Select Port 3 Bit n**[1)] 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for MLI I/O port control.

P4_PUDSEL
Port 4 Pull-Up/Pull-Down Select Register

Reset Value: 0000 00FF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								P7	P6	P5	P4	P3	P2	P1	P0
r								rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-7)**	n	rw	**Pull-Up/Pull-Down Select Port 4 Bit n** 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:8]	r	**Reserved;** read as 0; should be written with 0.

P0_PUDEN
Port 0 Pull-Up/Pull-Down Enable Register

Reset Value: 0000 FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 8-15)**	n	rw	**Pull-Up/Pull-Down Enable at Port 0 Bit n**[1)] 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for ASC I/O port control.

P3_PUDEN
Port 3 Pull-Up/Pull-Down Enable Register

Reset Value: 0000 FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 8-15)**	n	rw	**Pull-Up/Pull-Down Enable at Port 3 Bit n**[1)] 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for MLI I/O port control.

P4_PUDEN
Port 4 Pull-Up/Pull-Down Enable Register

Reset Value: 0000 00FF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								P7	P6	P5	P4	P3	P2	P1	P0
r								rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-7)**	n	rw	**Pull-Up/Pull-Down Enable at Port 4 Bit n** 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:8]	r	**Reserved;** read as 0; should be written with 0.

27.3.4 MLI0/MLI1 Register Address Ranges

In the TC1130, the registers of the MLI modules are located in the following address ranges:

- MLI0 module: Module Base Address = F010 $C000_H$
  Module End Address = F010 $C0FF_H$
  Small Transfer Windows (8 Kbytes max.)
  Pipe 0 = F01E 0000_H - F01E $1FFF_H$
  Pipe 1 = F01E 2000_H - F01E $3FFF_H$
  Pipe 2 = F01E 4000_H - F01E $5FFF_H$
  Pipe 3 = F01E 6000_H - F01E $7FFF_H$
  Large Transfer Windows (64 Kbytes max.)
  Pipe 0 = F020 0000_H - F020 $FFFF_H$
  Pipe 1 = F021 0000_H - F021 $FFFF_H$
  Pipe 2 = F022 0000_H - F022 $FFFF_H$
  Pipe 3 = F023 0000_H - F023 $FFFF_H$
- MLI1 module: Module Base Address = F010 $C100_H$
  Module End Address = F010 $C1FF_H$
  Small Transfer Windows (8 Kbytes max.)
  Pipe 0 = F01E 8000_H - F01E $9FFF_H$
  Pipe 1 = F01E $A000_H$ - F01E $BFFF_H$
  Pipe 2 = F01E $C000_H$ - F01E $DFFF_H$
  Pipe 3 = F01E $E000_H$ - F01E $FFFF_H$
  Large Transfer Windows (64 Kbytes max.)
  Pipe 0 = F024 0000_H - F024 $FFFF_H$
  Pipe 1 = F025 0000_H - F025 $FFFF_H$
  Pipe 2 = F026 0000_H - F026 $FFFF_H$
  Pipe 3 = F027 0000_H - F027 $FFFF_H$
- Absolute Register Address = Module Base Address + Offset Address
  (offset addresses see **Table 27-26**)

Note: The complete and detailed address map of the MLI0/MLI1 modules is described in the chapter “Register Overview” of the TC1130 System Units User’s Manual.

28 General Purpose Timer Unit (GPTU)

This chapter describes the General Purpose Timer Units GPTU of the TC1130. The information is presented in the following sections:

- Functional description of the GPTU Kernel (see **Section 28.1**)
- Register descriptions of all GPTU Kernel specific registers (see **Section 28.2**)
- TC1130 implementation specific details and registers of the GPTU (port connections and control, interrupt control, address decoding, clock control, see **Section 28.3**).

*Note: The GPTU kernel register names described in **Section 28.2** will be referenced in other parts of the TC1130 User's Manual with the module name prefix "GPTU_" for the GPTU module.*

28.1 GPTU Kernel Description

Figure 28-1 shows all the functional blocks of the GPTU kernel and its interfaces.

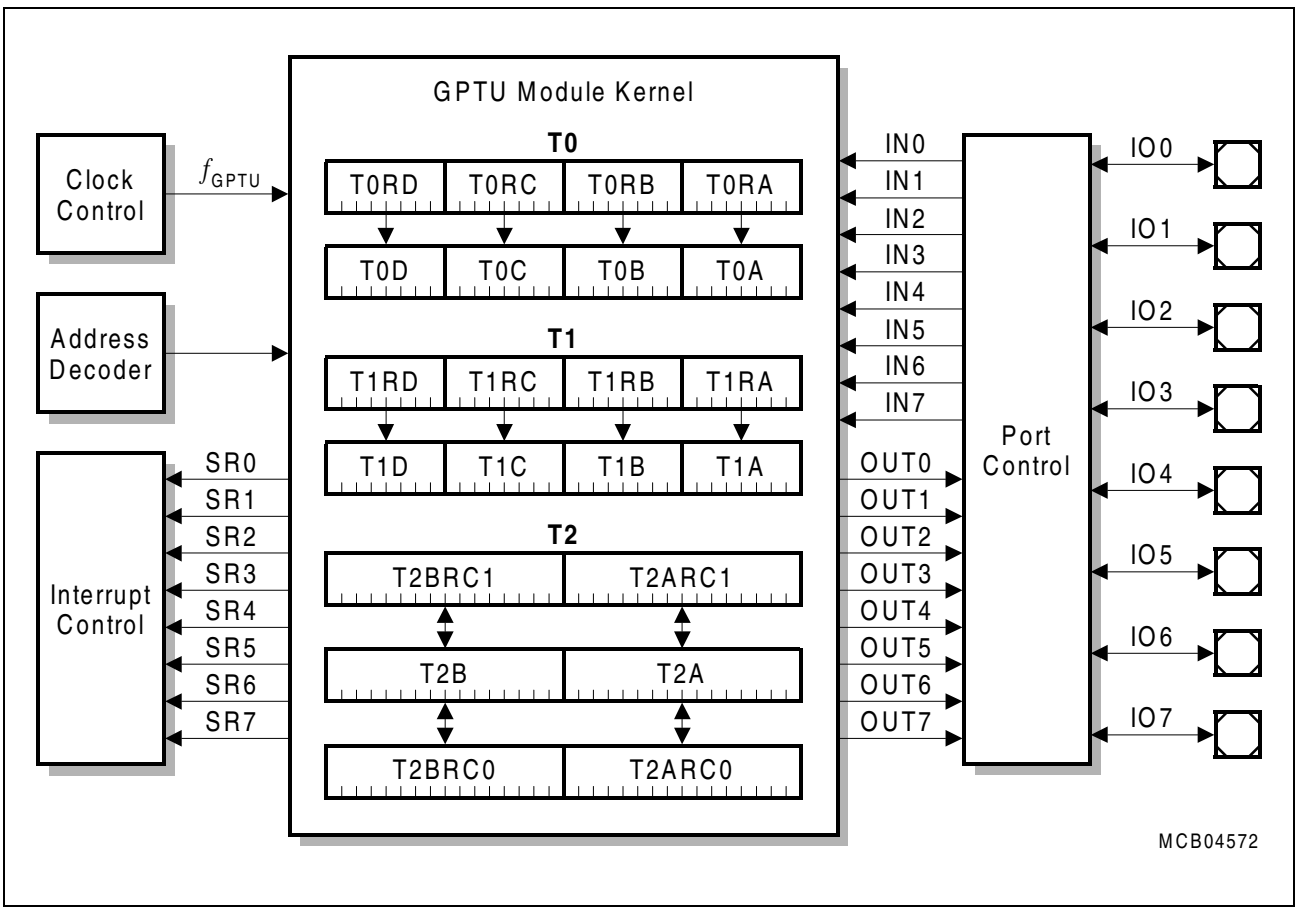


Figure 28-1 General Block Diagram of the GPTU Interface

The GPTU consists of three 32-bit timers designed to solve such application tasks as event timing, event counting, and event recording. The GPTU communicates with the external world via eight inputs and eight outputs concatenated in the port control logic to eight I/O pins IO[7:0]. The input signals coming from the port logic are named IN[7:0], and the output signals going to the port logic are named OUT[7:0]. These signals are used in the further descriptions of the timers. Further, the GPTU can generate eight service requests SR[7:0] within the TC1130.

Clock control, address decoding, and interrupt service request control are managed outside the GPTU Module kernel.

28.1.1 Operational Overview

The GPTU consists of the timers T0, T1, and T2. The functionality of the timers T0 and T1 differs from that of T2.

Additional features of timers T0 and T1 include:

- Each timer has a dedicated 32-bit reload register with automatic reload on overflow
- Timers can be split into individual 8-, 16-, or 24-bit timers with individual reload registers
- T0 and T1 can be concatenated to form one 64-bit timer
- Events generated in T0 or T1 can be used to trigger actions in T2
- Overflow signals can be selected to generate service requests, pin output signals, and T2 trigger events
- Two input pins can define a count option

Additional features of Timer T2 include:

- Up or down Count
- Operating modes:
  - Timer
  - Counter
  - Quadrature counter (incremental/phase encoded counter interface)
  - Options:
  - External start/stop, one-shot operation, timer clear on external event
  - Count direction control through software or an external event
  - Two 32-bit reload/capture registers
- Reload modes:
  - Reload on overflow or underflow
  - Reload on external event: positive transition, negative transition, or both transitions
- Capture modes:
  - Capture on external event: positive transition, negative transition, or both transitions
  - Capture and clear timer on external event: positive transition, negative transition, or both transitions
- Can be split into two 16-bit counter/timers
- Timer count, reload, capture, and trigger functions can be assigned to input pins. T0 and T1 overflow events can also be assigned to these functions
- Overflow and underflow signals can be used to trigger T0 and/or T1 and to toggle output pins
- T2 events are freely assignable to the service request nodes

28.1.2 Functional Overview

28.1.2.1 Timers T0 and T1

Figure 28-2 and Figure 28-3 show detailed block diagrams of Timers T0 and T1. Both, T0 and T1, consist of four 8-bit timer blocks named TxA, TxB, TxC, and TxD (x = 0, 1). Each eight-bit timer block contains a count register and a reload register. These blocks can be configured to run independently as 8-bit timers, or can be concatenated to form wider timers (16-bit, 24-bit, or 32-bit). A cross-connection between T0 and T1 extends these options to permit creation of a 64-bit timer.

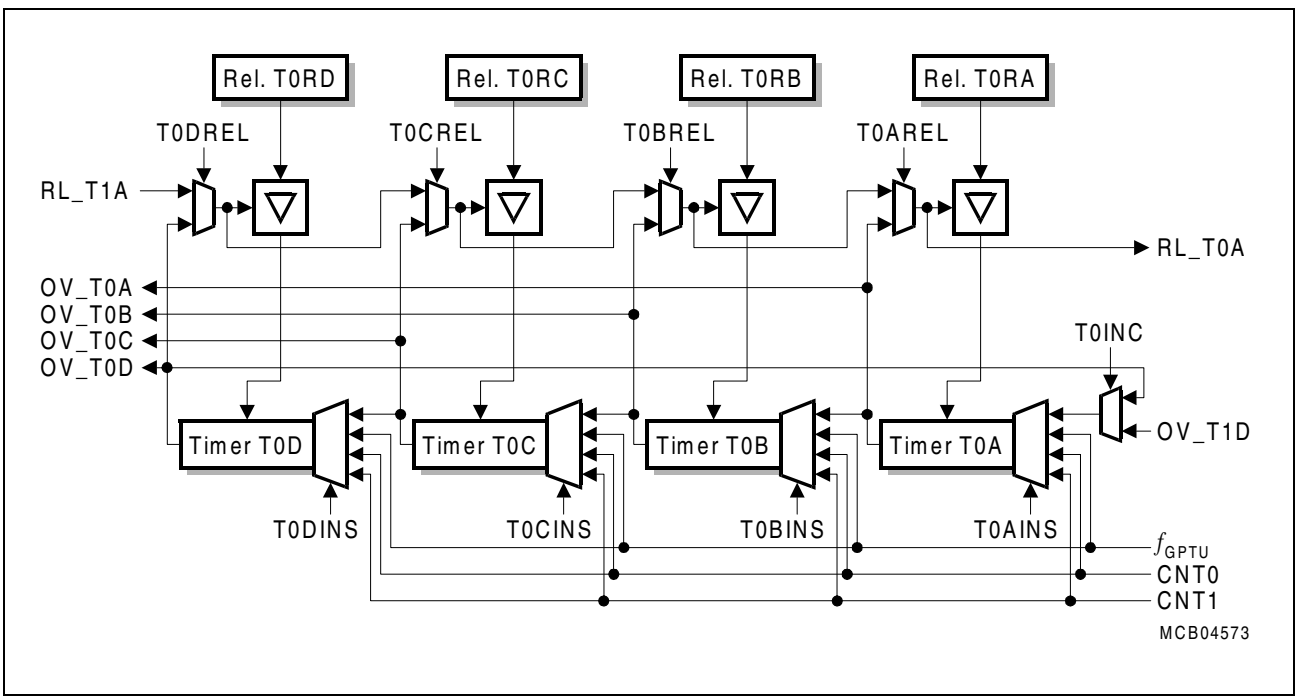


Figure 28-2 Detailed Block Diagram of T0

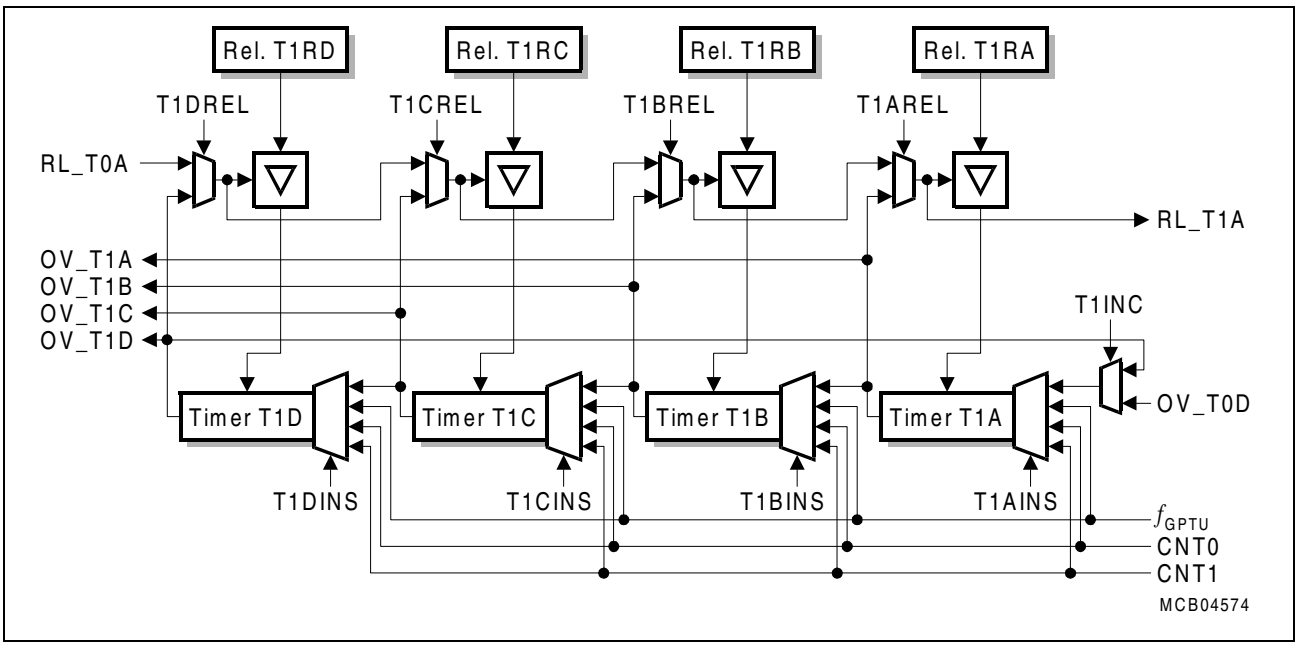


Figure 28-3 Detailed Block Diagram of T1

28.1.2.2 Input Selection

Each 8-bit timer block can select one of three possible inputs:

- The overflow of the previous timer (handled specially for T0A and T1A)
- An input frequency f_{GPTU} derived from the system clock
- One of two count inputs (CNT0, CNT1)

As shown in **Figure 28-2** and **Figure 28-3**, each of the four 8-bit timer blocks within T0 and T1 receives an overflow from the previous 8-bit timer block. Additionally, the A blocks of both timers can be separately configured to receive overflow either from its own D block, or the other's D block (by way of T0INC and T1INC). The two selectable configurations are:

1. The A blocks receive the overflow of their own D-block timer (T0A input is T0D overflow, and T1A input is T1D overflow).
2. The A blocks receive the overflow of the other's D-block timer (T0A input is T1D overflow, and T1A input is T0D overflow).

When configuration 1 is selected, T0 and T1 operate independently. Both timers can be set up individually as 8-bit, 16-bit, 24-bit, or 32-bit timers.

When configuration 2 is selected T0 and T1 inter-operate, and can be concatenated to form wider timers. For 40-bit, 48-bit, 56-bit or 64-bit operation, the timer not receiving overflow from the other timer must be driven by the module clock, CNT0, or CNT1. Additionally, the overflow selection of the other 8-bit timers within T0 and T1 must all be configured appropriately to source overflow from its previous timer.

The source for the two count inputs (CNT0 or CNT1) can be either an external input or a trigger signal from T2 (by way of T2 overflow signals, OUV_T2A and OUV_T2B). **Figure 28-4** shows these options.

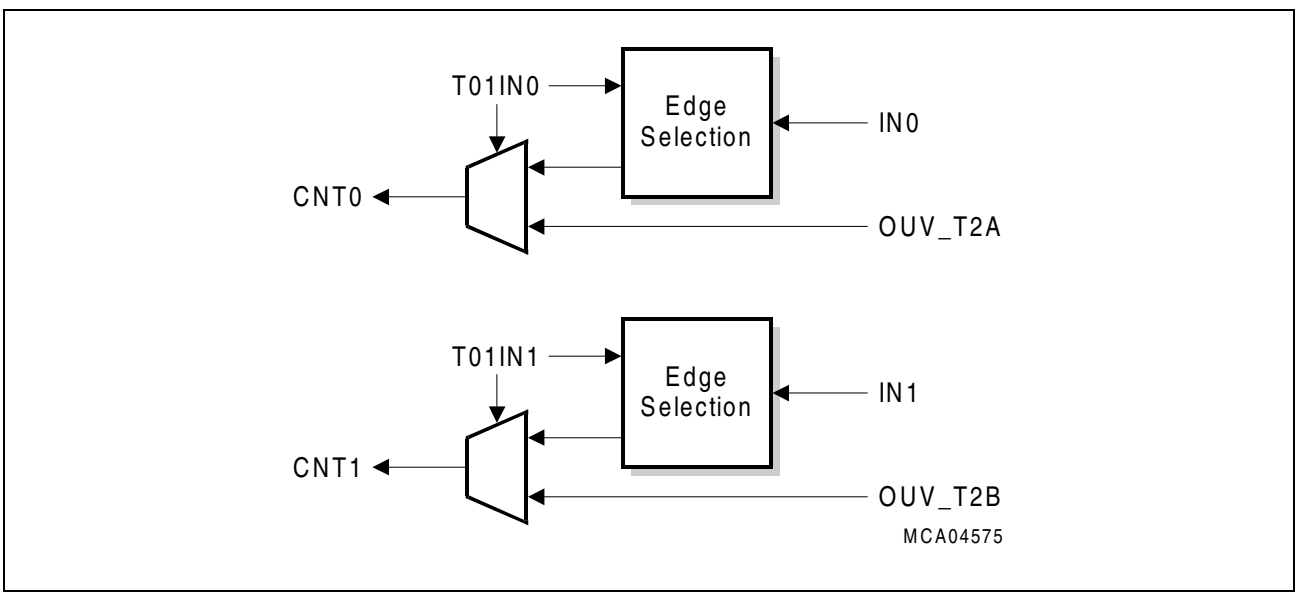


Figure 28-4 Timer T0 and T1 Global Input Control

Access to Timer T0 and T1 Count and Reload Registers

Two address locations are provided for each of the count and reload registers, which enable access to the appropriate registers even for a 24-bit timer configuration. The first address location provides all four bytes of a timer count/reload register. The symbolic name for this address indicates that all four parts, D … A, are accessible. Registers TxDCBA provide access to the count registers and registers TxRDCBA provide access to the reload registers. Individual access to the single bytes, combined 16-bit half-word-aligned combination, or full 32-bit combination, is possible in this way. The second address location provides the lower three bytes of a timer count/reload register; the most significant byte is not connected. The second address location enables access to a timer count/reload register in a 24-bit combination without corrupting the upper byte of the timer count/reload register. The symbolic name for this second address location is TxCBA (for the count registers) and TxRCBA (for the reload registers). These locations provide access only to the lower three parts, C … A, of the timer count and reload registers. **Table 28-1** gives an overview on the different access options to the individual combinations of T0 and T1.

Table 28-1 Access Options to T0/T1

Register	Access Width	Least Significant Address Bits	TxD TxRD	TxC TxRC	TxB TxRB	TxA TxRA
TxDCBA TxRDCBA	Byte	000				x
	Byte	001			x	
	Byte	010		x		
	Byte	011	x			
	Half-word	000			x	
	Half-word	010	x			
	Word	000	x			
TxCBA TxRCBA	Byte	100				x
	Byte	101			x	
	Byte	110		x		
	Byte	111	0			
	Half-word	100			x	
	Half-word	110	0	x		
	Word	100	0	x		

Reading and writing to the individual byte or half-word parts of a timer is performed on the first address location using byte or half-word load/store operations. The entire 32-bit timer is accessed with word load/store operations. Reading from the second address location with a word load operation provides the contents of the lower three bytes of the timer count/reload register, with the most significant byte returning 0. Writing to it with a word store operation affects only the lower three bytes. The value of the most significant byte is not stored. It is recommended that software always writes 0 to the most significant byte. The second address location can also be accessed with byte or half-word load/store operations.

Note: Access to a 16-bit half-word that crosses a half-word boundary (for example, the combination of T0C and T0B as one 16-bit timer) and access to a 24-bit combination using the upper three bytes (for example, T0D, T0C, and T0B) are not provided. Because it is always possible to align 16-bit timers on half-word boundaries, and right-align a 24-bit timer, these combinations are not required.

28.1.2.3 Reload Selection

As shown in **Figure 28-2** and **Figure 28-3**, the reload trigger signals for the reload registers are controlled independently from timer concatenation. The independent control provides the option of concatenating timers while giving each timer its own reload

period. Reload selection is controlled by T0xREL and T1xREL so that each eight-bit timer can be triggered by either:

- The overflow of its own counter
- The reload event of one of the higher-order timer(s)

28.1.2.4 Service Requests, Output Signals, and Trigger Signals

Overflow signals from T0 and T1 can be used to generate service requests, output signals, or trigger signals for T2.

The four overflow signals from each 8-bit timer in T0 and T1 can trigger two service requests, two output signals, and two trigger signals. These options are shown in **Figure 28-5** for T0 and **Figure 28-6** for T1.

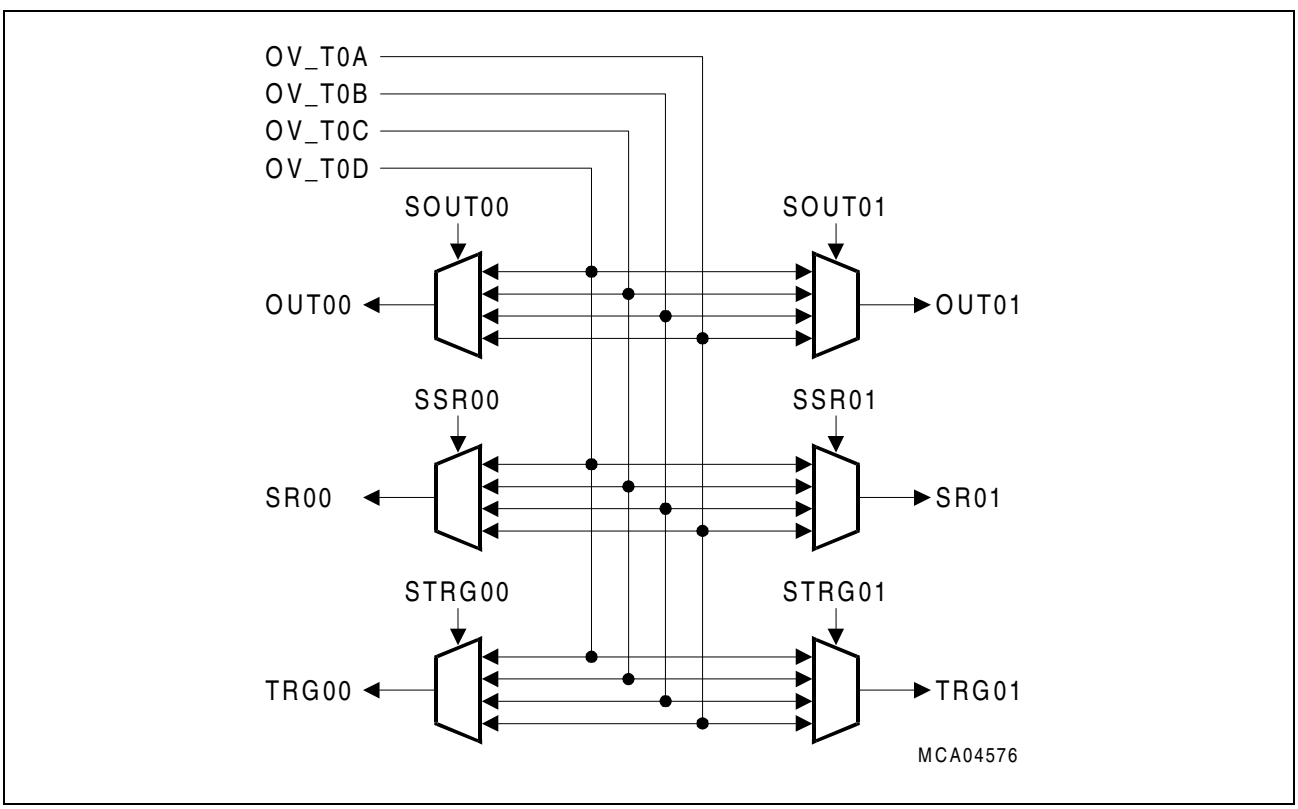


Figure 28-5 Timer T0 Output, Trigger, and Service Request Selection Control

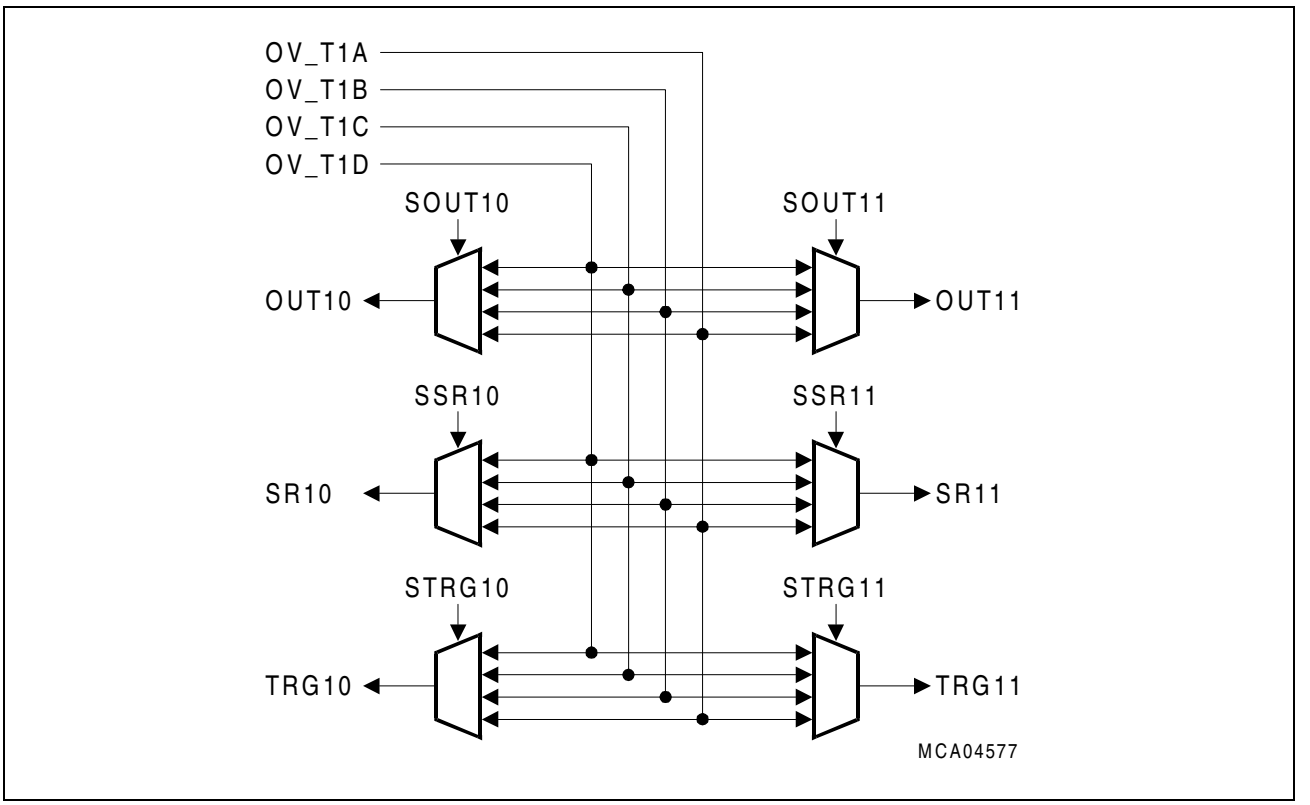


Figure 28-6 Timer T1 Output, Trigger, and Service Request Selection Control

28.1.2.5 Timers T0 and T1 Configuration Limitations

Due to timing delays of the internal circuitry, there are certain special cases and restrictions associated with the configuration possibilities of Timers T0 and T1.

In the following cases, one additional GPTU clock pulse is inserted into the count or reload signal:

- Overflow of T0D is used as count input to T0A
- Overflow of T1D is used as count input to T0A
- Overflow of T1D is used as count input to T0A
- Reload trigger of T0A/T0RA is used as reload trigger for T1D/T1RD

These combinations should either be avoided or the additional clock pulse needs to be taken into account. In the first three cases, if the timer producing the overflow is used as a prescaler for the following timer, the effect of the additional clock pulse is usually irrelevant. The prescaler just needs to be started such that the timer contents are one count higher than the reload value. This avoids a longer initial period due to the pulse delay.

It is recommended that the fourth case is always avoided. This case would occur if T0 and T1 (or parts of them) are concatenated such that T1D is the less significant and T0A is the more significant part of this timer combination. The overflow of T1D would be used as count input to T0A would experience a clock delay. The reload trigger line from T0A

back to T1D would experience another clock delay, resulting in a total delay of two GPTU clocks from T1D overflow to its reload event. Because T1D continues counting after its overflow, its contents will be overwritten by the reload two clock cycles later, resulting in the loss of two counts.

Concatenating T0 and T1 such that T0 contains the less significant part of the combined timer does not present a problem. The overflow of T0D to T1A and the reload trigger signal from T1A back to T0D do not have this extra delay.

Due to the high flexibility of the configuration options for Timers T0 and T1, it is almost never required to use one of the cases described above.

28.1.2.6 Timer T2

Timer T2 consists of two 16-bit timer blocks, T2A and T2B. Each 16-bit timer block contains a count register and two reload/capture registers. These blocks can be configured to form one 32-bit timer as shown in **Figure 28-7**, or to run independently as two 16-bit timers as shown in **Figure 28-8**. This basic configuration of Timer T2 is controlled by the T2CON.T2SPLIT control bit.

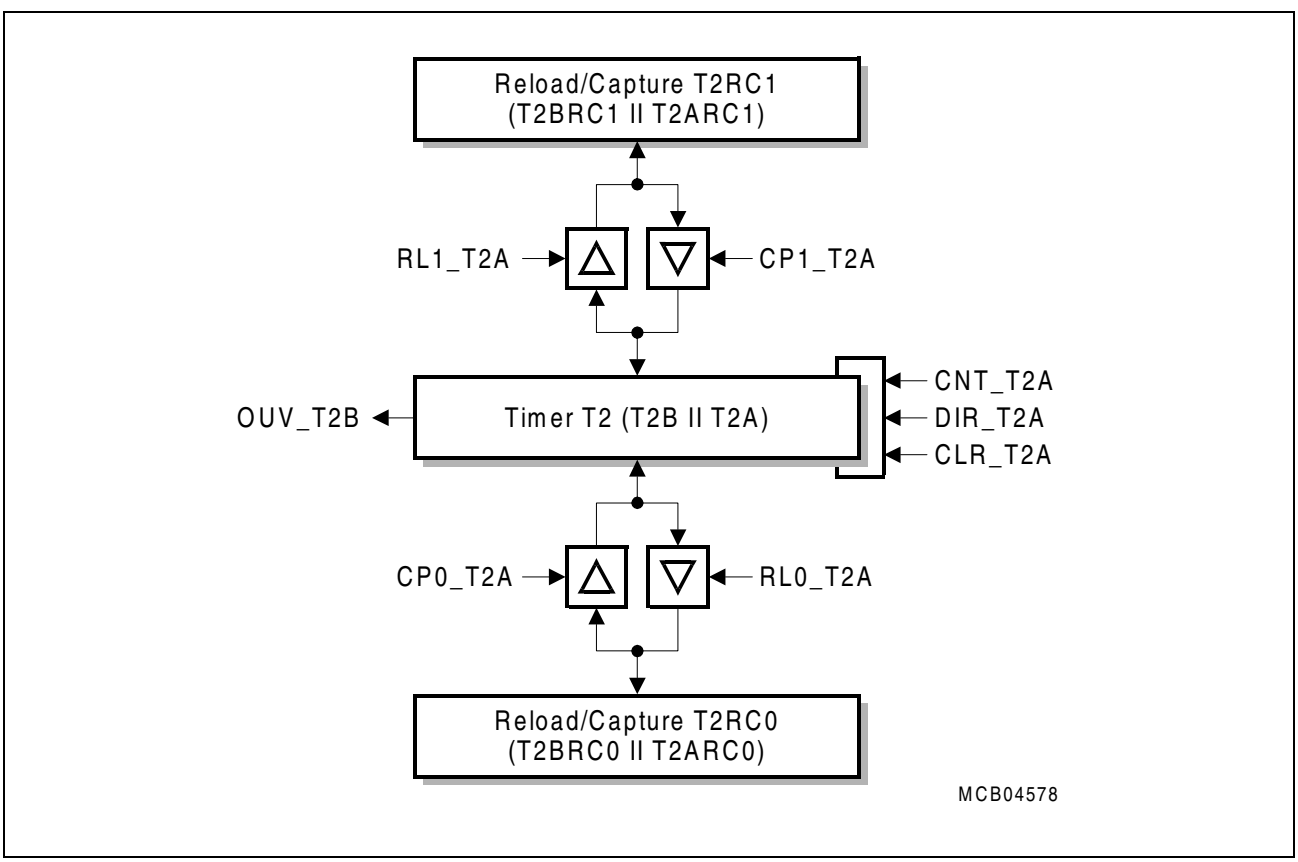


Figure 28-7 Block Diagram of Timer 2 in 32-Bit Mode

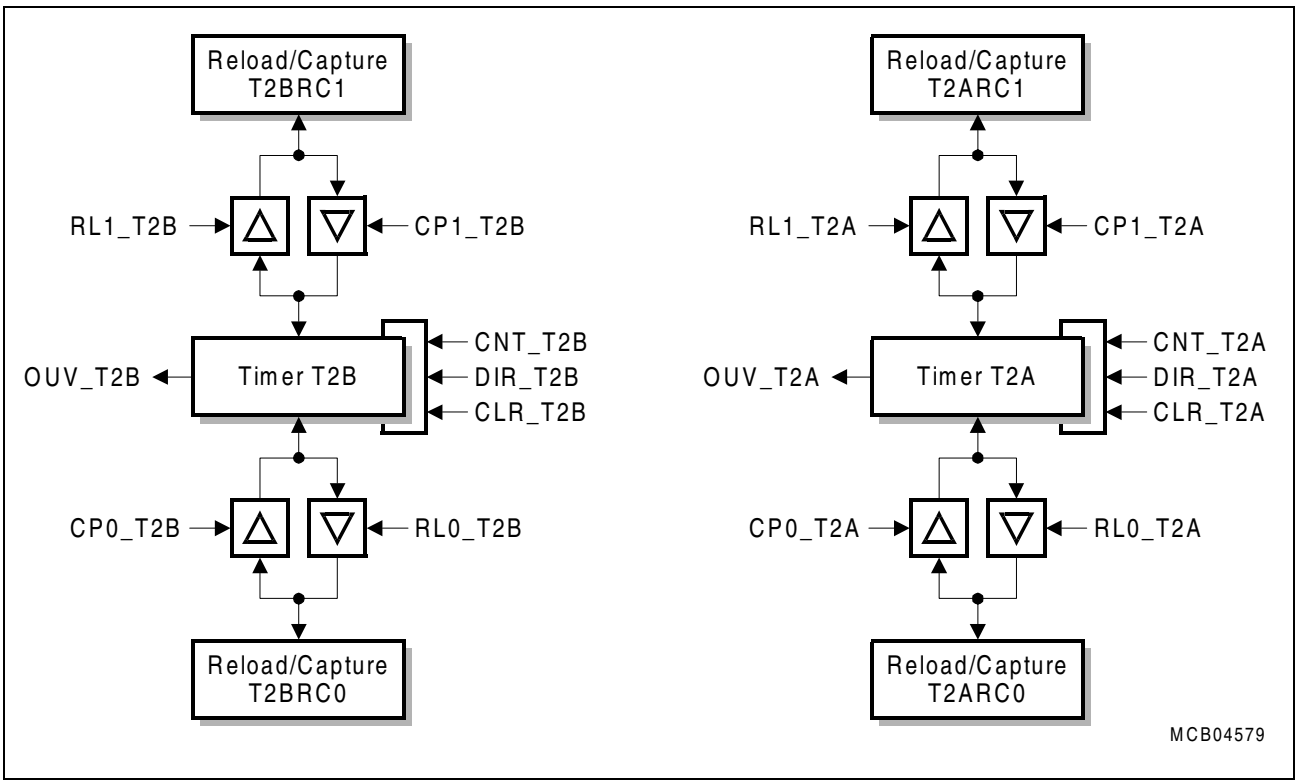


Figure 28-8 Block Diagram of Timer 2 in Split Mode

As shown in **Figure 28-9**, any of the eight GPTU input lines can be assigned to trigger any of the functions performed by T2, including count, start, stop, change direction, clear, reload/capture, and service request. Each of these functions can be selectively triggered on a positive edge, a negative edge, or both edges of the input signal. In addition to these external inputs, signals from Timers T0 and T1 can be used to trigger functions in T2.

All external inputs can be assigned to any of the input functions of T2A and T2B, whether they are split or concatenated. When concatenated, all functions in T2A and T2B are controlled by the T2A mode control block. When split, T2A and T2B are controlled by their individual mode control blocks.

Three registers select the input line and the triggering edge for a specific function. The first register, T2AIS, selects the inputs for either T2 in 32-bit mode or T2A in Split Mode. Register T2BIS does the same for T2B in Split Mode. The third register, T2ES, provides the means to select which edge of the selected external signal causes a trigger of the associated function.

Most of these input signals can be used to generate a service request, independent of whether they are used to trigger Timer T2 functions or not.

Two registers control the mode of operation for the timer and the reload/capture registers. They also provide status information. Register T2CON controls the operation of the timer itself and holds the status information. Register T2RCCON controls the operation of the two reload/capture registers.

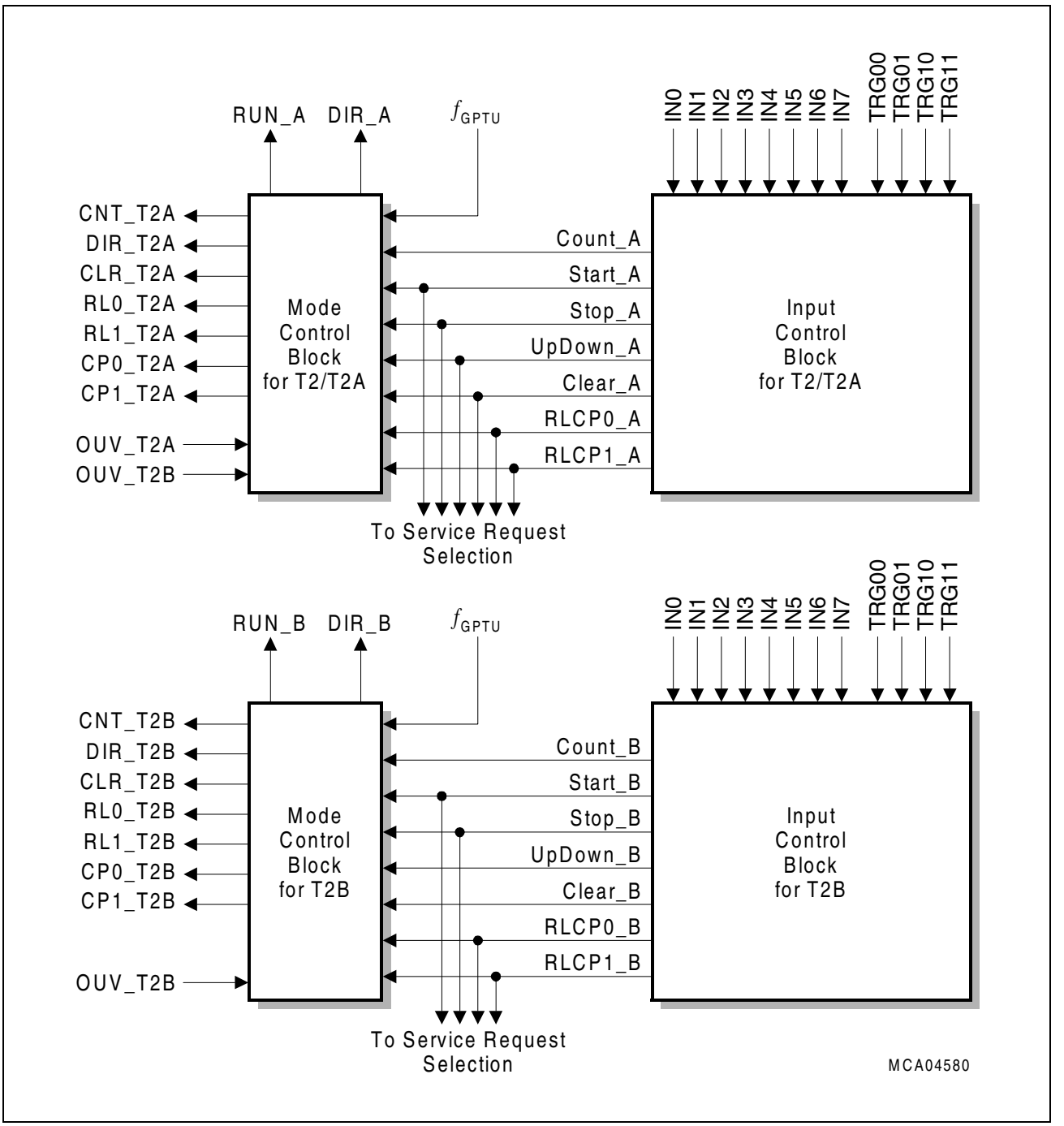


Figure 28-9 Timer 2 Input and Mode Control Blocks

Figure 28-10 and **Figure 28-11** show how T2 control signals are determined. This information is summarized as follows.

- Count control CNT_T2x
  - Clock Source Control, T2CON.T2xCSRC, determines the clocking trigger. Input can be the module clock (f_{GPTU}), or an external trigger source, Count_x. In Quadrature Counter Mode, count input sources are the two inputs, Count_x and UpDown_x.
  - External clocking trigger, Count_x, is determined by T2xIS.T2xICNT. Trigger source can be either an external input, INy, or a trigger signal, TRGxx, from Timer T0 or Timer T1. Bit T2ES.T2xECNT determines the active clock edge.
  - Starting and stopping of the timer can be controlled either by software via setting or clearing the run bit T012RUN.T2xRUN (software modifications of this bit are performed through the run bit set and clear bits, T012RUN.T2xSETR and T012RUN.T2xCLRR, respectively), or through the signals Start_x and Stop_x, selected by T2xIS.T2xISTR and T2xIS.T2xISTP, respectively. Any external input INy can be selected for this purpose. T2ES.T2xESTR and T2ES.T2xESTP determine the active clock edges for these sources, respectively.
    Additionally, in one-shot mode, the timer is stopped in response to its own overflow, OUV_T2x.
  - The running/stopped status of T2A and T2B can be examined via the T012RUN.T2xRUN status bits.
- Count direction control DIR_T2x
  - Input source control, T2CON.T2xCDIR, selects whether the count direction is up or down, or whether it is determined from an external input.
  - External input selection is controlled by T2xIS.T2xIUD, which selects any of the INy input signals. T2ES.T2xEUD determines the active clock edge. In Quadrature Counter Mode, up/down count information is derived from the two input sources, Count_x and UpDown_x.
- Clear control CLR_T2x
  - T2CON.T2xCCLR selects whether to clear the timer to 0 on an external event (Clear_x), or to clear the timer on capture 0 event (CP0_T2x), or to clear timer on capture 1 event (CP1_T2x).
  - Selection of the external trigger is determined by T2xIS.T2xICLR, which selects any of the INy input signals. T2ES.T2xECLR determines the active clock edge.
- Reload/capture RL0_T2x, RL1_T2x, and CP0_T2x, CP1_T2x
  - There are two reload/capture registers each in T2A and T2B which can be programmed independently.
  - Controls T2RCCON.T2xMRC0 and T2RCCON.T2xMRC1 determine reload/capture modes. Modes include disabled, capture on external event, reload on overflow or underflow, reload on external event, reload on overflow only, reload on underflow only, reload on external event if count direction is up (if

T2CON.T2xDIR = 0), reload on external event if count direction is down (T2CON.T2xDIR = 1).

– Selection of external trigger source for RLCP0_x and RLCP1_x is determined by T2xIS.T2xIRC0 and T2xIS.T2xIRC1. Trigger source can be either an external input, GPTUx_INy, or a trigger signal, TRGxx, from T0 or T1. T2ES.T2xERC0 and T2ES.T2xERC1 determine the active edge of the trigger signal.

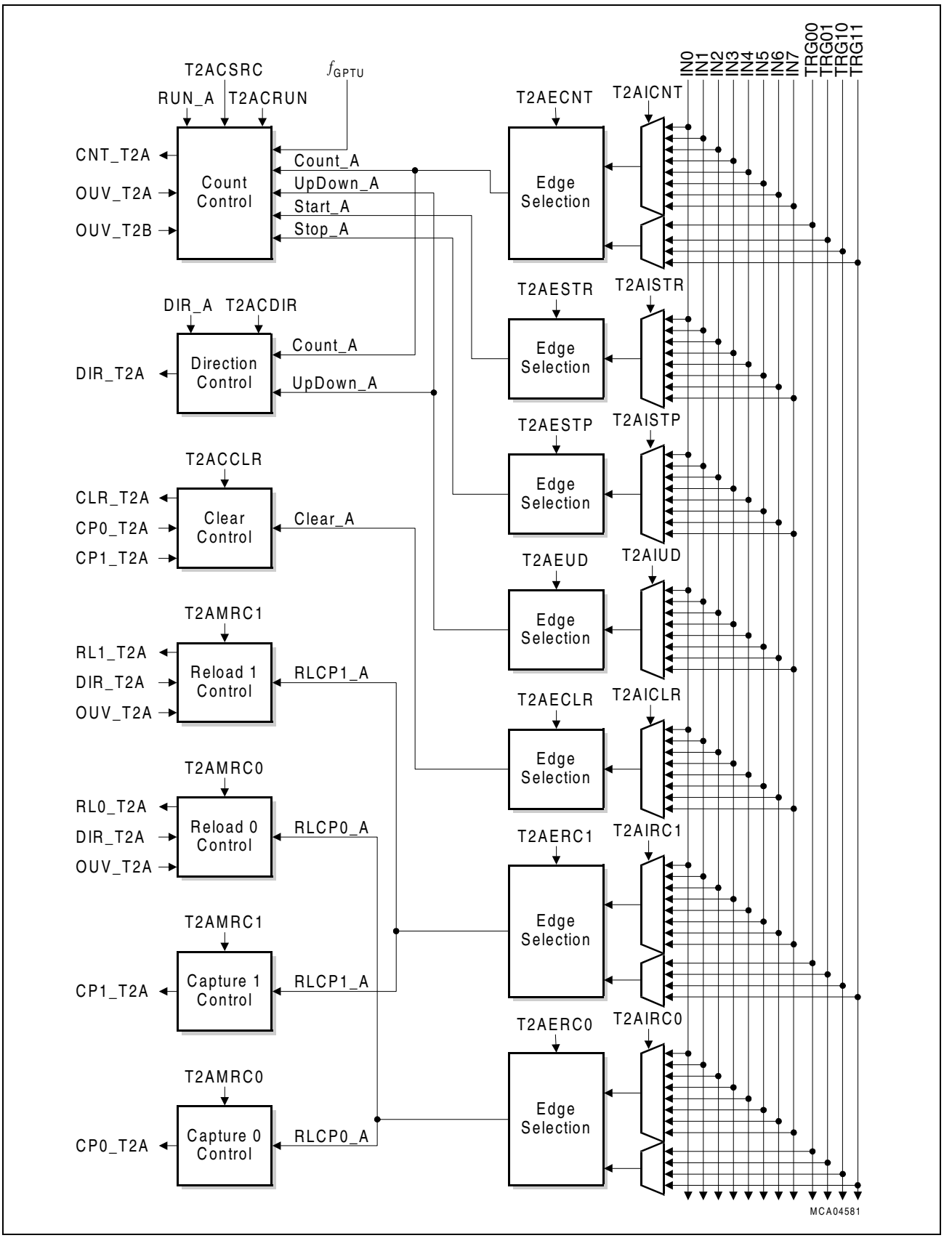


Figure 28-10 Timer T2/T2A Input and Mode Control Details

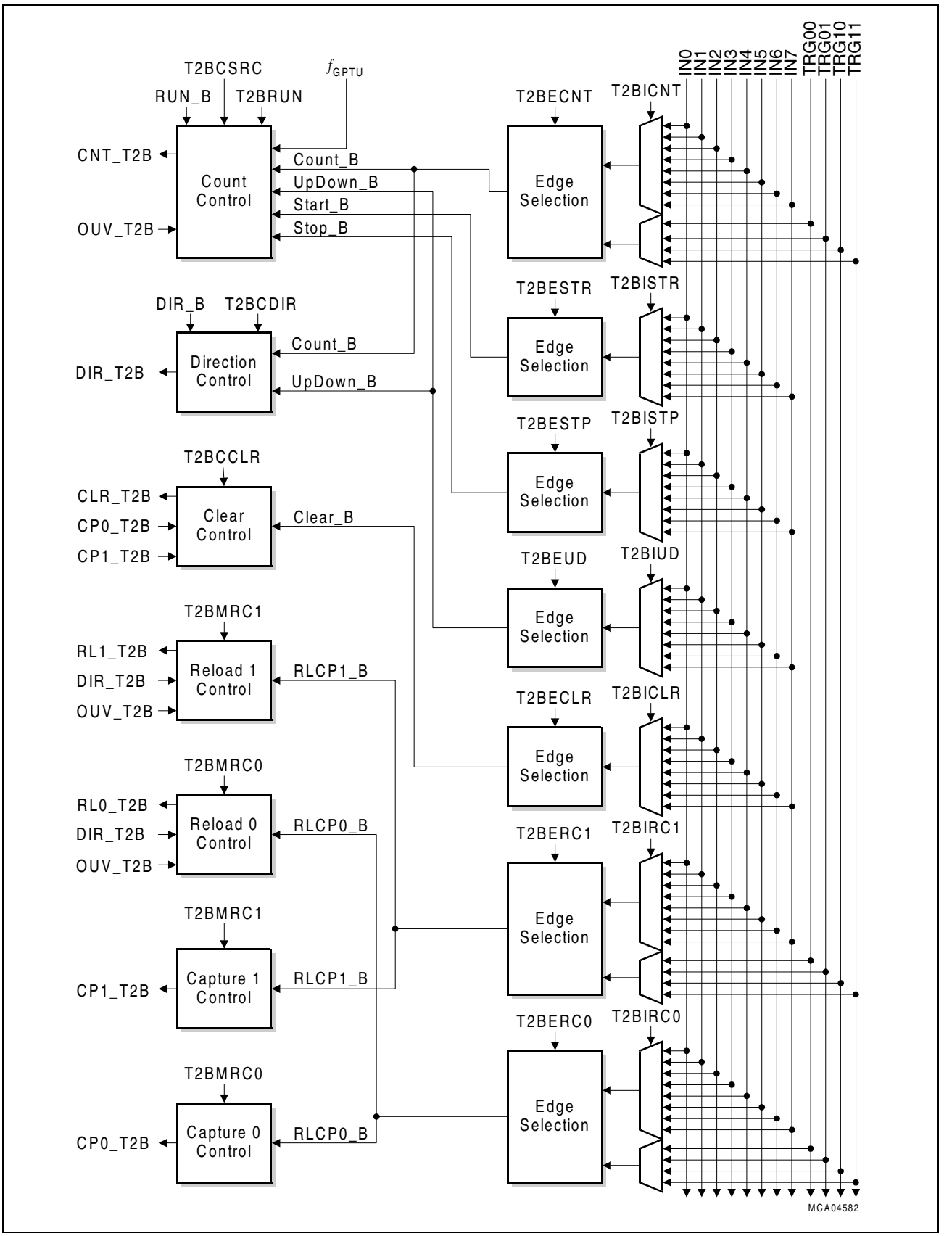


Figure 28-11 Timer T2B Input and Mode Control Details

28.1.2.7 Quadrature Counting Mode

Position tracking can be performed with Timer T2 in Quadrature Counting Mode, sometimes referred to as incremental or phase encoded interface. The standard way of tracking positions is to use two phase-shifted input signals. These provide the counting and direction information necessary for this task. As shown in **Figure 28-12**, the edges of the signals provide the count signal, while the phase relation between the two signals provides the direction information.

To operate Timer T2 in this mode, the two signals are connected such that they trigger the Count_A/Count_B and the UpDown_A/UpDown_B inputs of the timer block.

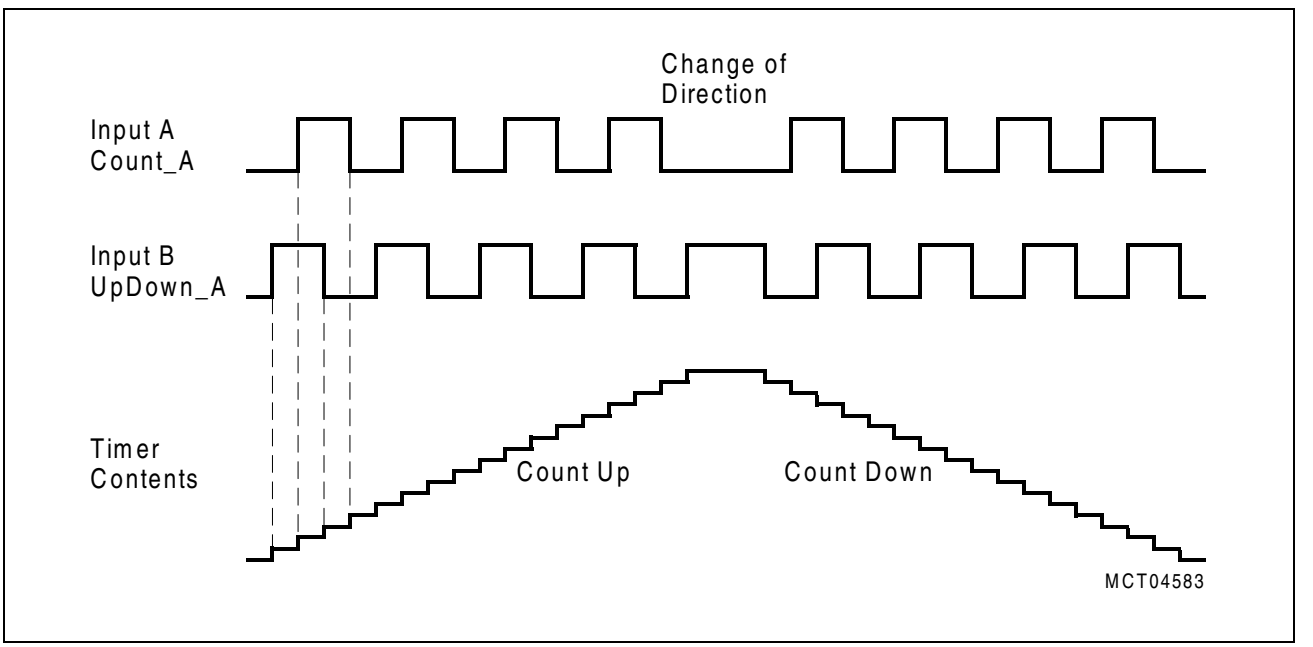


Figure 28-12 Quadrature Counting Operation

28.1.3 Global GPTU Controls

This section describes global control of the GPTU. Global controls are provided for the outputs and interrupt service requests.

28.1.3.1 Output Control

The register OUT has eight bits OUTx (x = 0-7) which store the output signals from the GPTU. The bits in register OUT can also be set or cleared via software. The connection of timer signals to these output bits is determined by eight bit fields in register OSEL, named SOx (x = 0-7). Each output bit in register OUT is connected to a GPTU output line, which connects to the Parallel Ports.

Six signals from Timers T0, T1, and T2 can be selected to generate outputs from the GPTU timers to the Parallel Ports. For each of the eight GPTU output signals, OUT[7:0], the user can select which of the timer signals, OUT00, OUT01, OUT10, OUT11, OUV_T2A, or OUV_T2B, activates the selected output line.

OUT00 and OUT01 can be any T0 timer overflow, OUT10, OUT11 can be any T1 timer overflow. OUV_T2A and OUV_T2B are the timer overflows of T2A and T2B.

Figure 28-13 provides an overview of the output options.

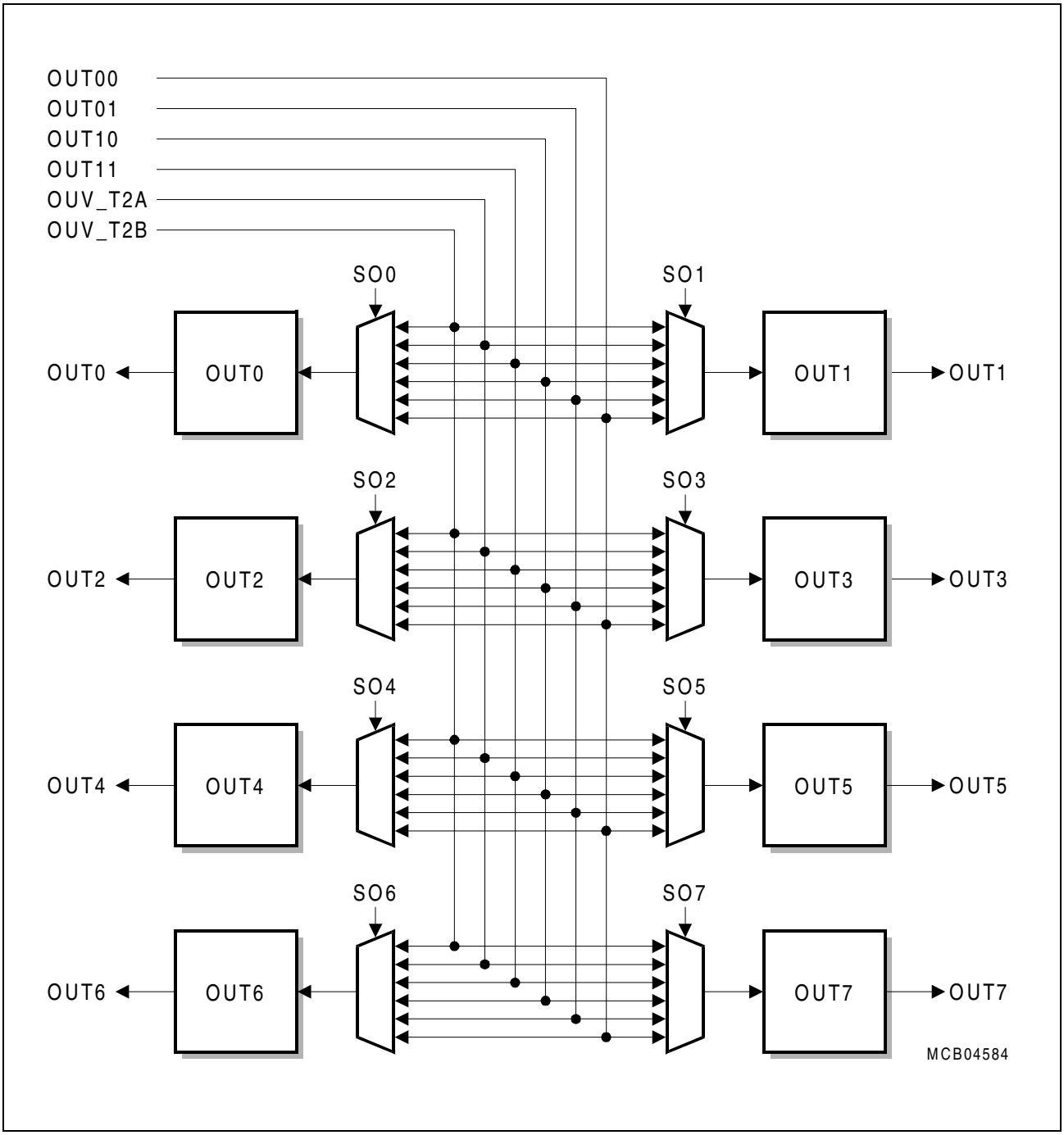


Figure 28-13 Output Control Block Diagram

28.1.3.2 Service Request Control

Sixteen events in T0, T1, and T2 can be selected to generate a service request to the CPU. Eight service request outputs (nodes), SR[7:0], are provided for the GPTU; they can be freely assigned to any of the GPTU events.

Timer T2 events which can be selected include Start_x, Stop_x, UpDown_A, Clear_A (signals UpDown_B and Clear_B are not available for service request generation), RLCP0_x, RLCP1_x, OUV_T2x. Timer T0 overflow events (SR00, SR01) and Timer T1 overflow events (SR10, and SR11) can also be selected. **Figure 28-14** shows these options. Please note that the signals Start_x, Stop_x, UpDown_A, Clear_A, RLCP0_x, and RLCP1_x are the signals coming out of the input selection block, before these lines go into the Timer T2 control logic (see **Figure 28-9**). This has the advantage, that an input line can be used to generate a service request only; it may or may not be used to also trigger a T2 function. In this way, all of the GPTU input lines connected to parallel port pins can be configured as external interrupt inputs.

Because Timers T0 and T1 can generate triggers for Timer T2 signals (such as Count_x, RLCP0_x, and RLCP1_x), it is possible to use these signals for service request generation (whether or not they are also used to trigger functions of T2). This gives additional service requests to Timers T0 and T1.

Because of the flexibility in selecting service requests, more than one service request can be generated by the same event.

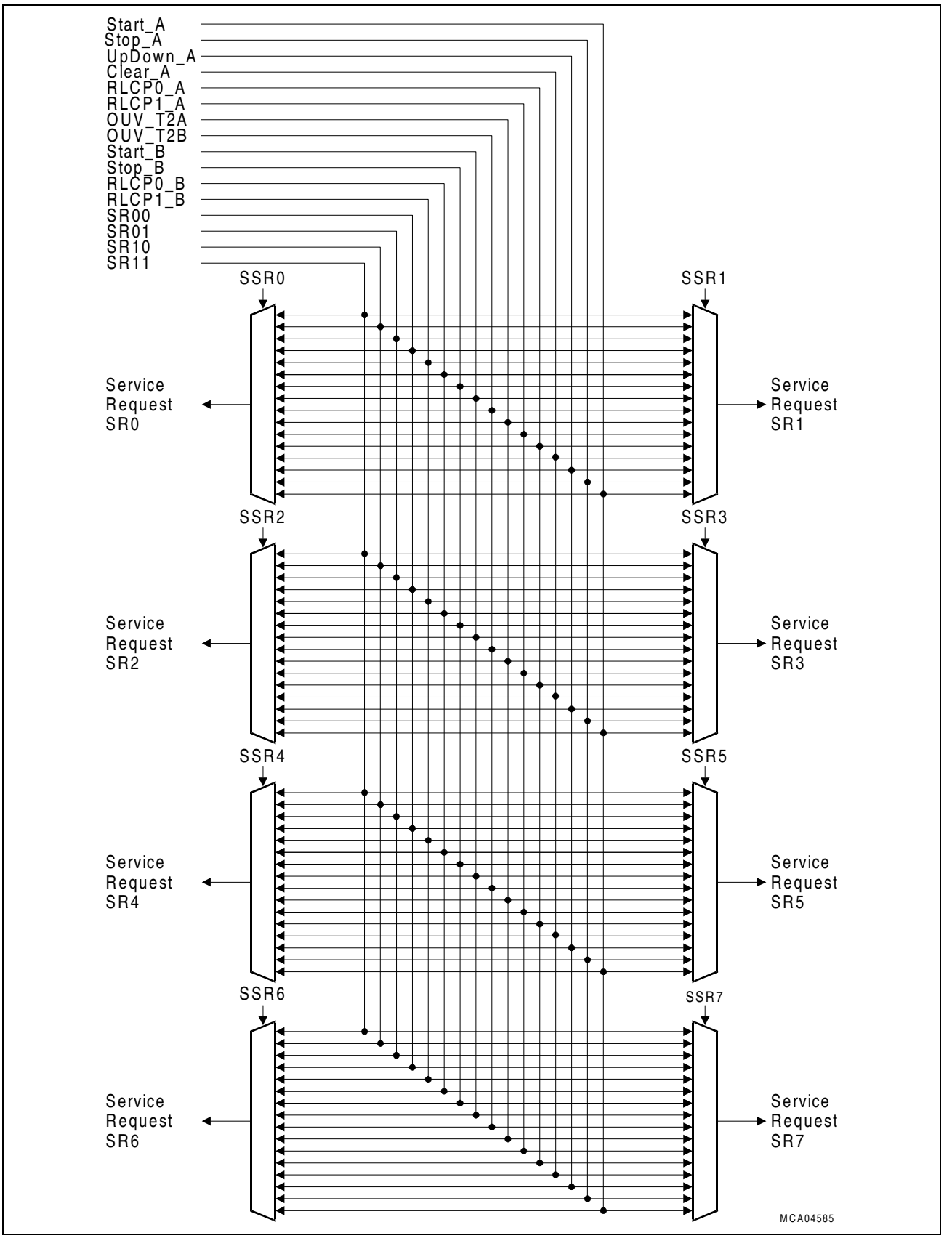


Figure 28-14 Service Request Selection

28.2 GPTU Kernel Registers

Figure 28-15 and Table 28-2 show all registers associated with the GPTU Kernel.

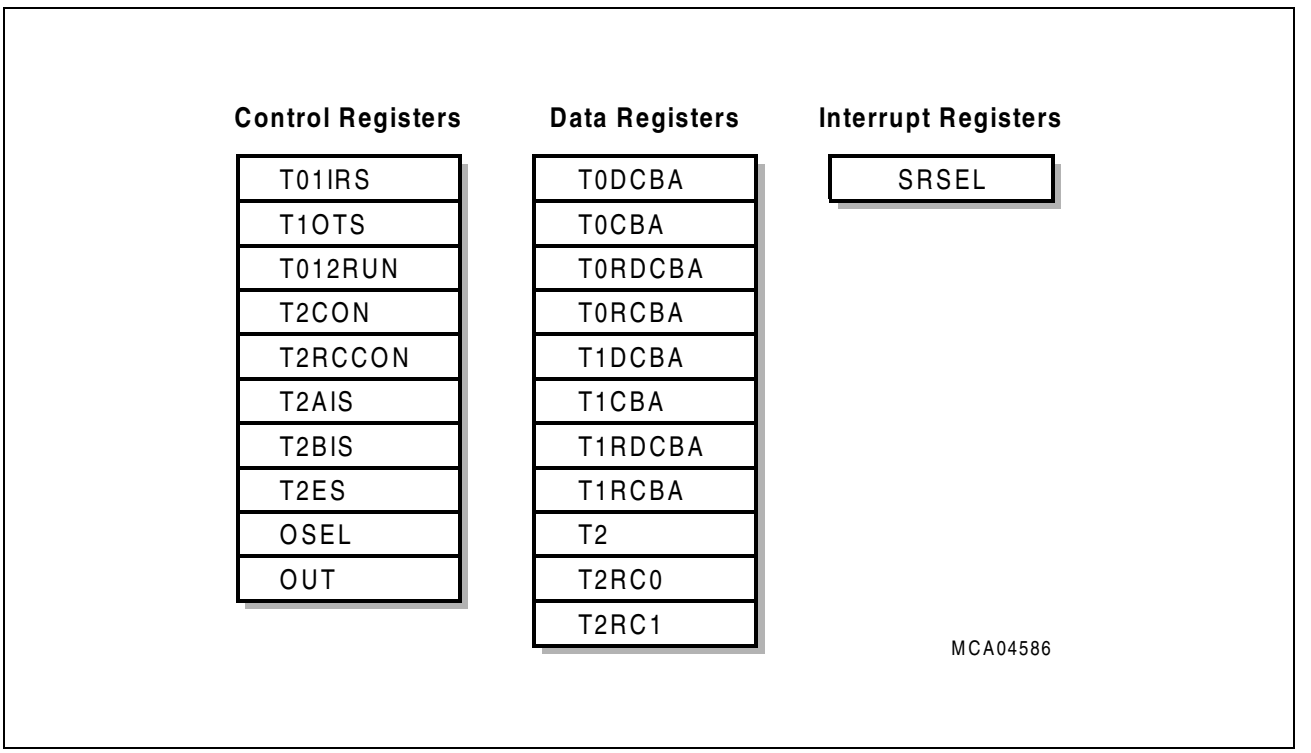


Figure 28-15 GPTU Kernel Registers

Table 28-2 GPTU Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
T01IRS	Timer T0 and T1 Input and Reload Source Selection Register	0010_H	Page 28-24
T01OTS	Timer T0 and T1 Output, Trigger and Service Request Register	0014_H	Page 28-27
T2CON	Timer T2 Control Register	0018_H	Page 28-38
T2RCCON	Timer T2 Reload/Capture Control Register	$001C_H$	Page 28-43
T2AIS	Timer T2/T2A Ext. Input Selection Register	0020_H	Page 28-33
T2BIS	Timer T2B External Input Selection Register	0024_H	Page 28-35
T2ES	Timer T2 External Input Edge Selection Reg.	0028_H	Page 28-36
OSEL	Output Source Selection Register	$002C_H$	Page 28-47
OUT	Output Register	0030_H	Page 28-48

Table 28-2 GPTU Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
T0DCBA	Timer T0 Count Register (T0D, T0C, T0B, T0A)	0034_H	Page 28-29
T0CBA	Timer T0 Count Register (T0C, T0B, T0A)	0038_H	Page 28-29
T0RDCBA	Timer T0 Reload Register (T0RD, T0RC, T0RB, T0RA)	$003C_H$	Page 28-30
T0RCBA	Timer T0 Reload Register (T0RC, T0RB, T0RA)	0040_H	Page 28-30
T1DCBA	Timer T1 Count Register (T1D, T1C, T1B, T1A)	0044_H	Page 28-31
T1CBA	Timer T1 Count Register (T1C, T1B, T1A)	0048_H	Page 28-31
T1RDCBA	Timer T1 Reload Register (T1RD, T1RC, T1RB, T1RA)	$004C_H$	Page 28-31
T1RCBA	Timer T1 Reload Register (T1RC, T1RB, T1RA)	0050_H	Page 28-32
T2	Timer T2 Count Register	0054_H	Page 28-45
T2RC0	Timer T2 Reload/Capture Register 0	0058_H	Page 28-46
T2RC1	Timer T2 Reload/Capture Register 1	$005C_H$	Page 28-46
T012RUN	Timers T0, T1, T2 Run Control Register	0060_H	Page 28-41
SRSEL	Service Request Source Select Reg.	$00DC_H$	Page 28-49

28.2.1 Timer T0/T1 Registers

This section describes the registers related to Timers T0 and T1. Note that register T012RUN is shared between all three timers and is described in Section 28.2.2.3.

28.2.1.1 Timer T0/T1 Input & Reload Source Selection Register

The T01IRS register contains the individual controls for the count input and the reload trigger selections for the individual parts of T0 and T1. This register also contains the control for the global input signals CNT0 and CNT1.

T01IRS
Timer T0 and T1 Input and Reload Source Selection Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
T01 IN1		T01 IN0		0		T1 INC	T0 INC	T1D REL	T1C REL	T1B REL	T1A REL	T0D REL	T0C REL	T0B REL	T0A REL
rw		rw		r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T1D INS		T1C INS		T1B INS		T1A INS		T0D INS		T0C INS		T0B INS		T0A INS	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
T0AINS	[1:0]	rw	**T0A Input Selection** 00 Clock input f_{GPTU} 01 Global input CNT0 10 Global input CNT1 11 Carry input (concatenation)
T0BINS	[3:2]	rw	**T0B Input Selection;** coding as T0AINS
T0CINS	[5:4]	rw	**T0C Input Selection;** coding as T0AINS
T0DINS	[7:6]	rw	**T0D Input Selection;** coding as T0AINS
T1AINS	[9:8]	rw	**T1A Input Selection;** coding as T0AINS
T1BINS	[11:10]	rw	**T1B Input Selection;** coding as T0AINS
T1CINS	[13:12]	rw	**T1C Input Selection;** coding as T0AINS
T1DINS	[15:14]	rw	**T1D Input Selection;** coding as T0AINS

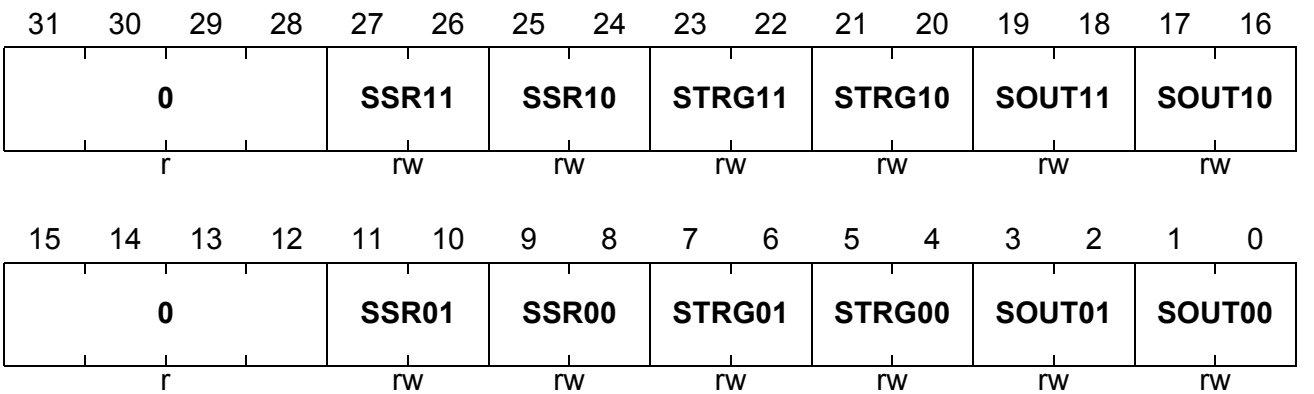
Field	Bits	Type	Description
T0AREL	16	rw	**T0A Reload Source Selection** 0 Reload on overflow of timer T0A 1 Concatenation with T0RB
T0BREL	17	rw	**T0B Reload Source Selection** 0 Reload on overflow of timer T0B 1 Concatenation with T0RC
T0CREL	18	rw	**T0C Reload Source Selection** 0 Reload on overflow of timer T0C 1 Concatenation with T0RD
T0DREL	19	rw	**T0D Reload Source Selection** 0 Reload on overflow of timer T0D 1 Reload on signal T1RA
T1AREL	20	rw	**T1A Reload Source Selection** 0 Reload on overflow of timer T1A 1 Concatenation with T1RB
T1BREL	21	rw	**T1B Reload Source Selection** 0 Reload on overflow of timer T1B 1 Concatenation with T1RC
T1CREL	22	rw	**T1C Reload Source Selection** 0 Reload on overflow of timer T1C 1 Concatenation with T1RD
T1DREL	23	rw	**T1D Reload Source Selection** 0 Reload on overflow of timer T1D 1 Concatenation with T0RA
T0INC	24	rw	**T0 Carry Input Selection** 0 T0A carry in is T0D carry out 1 T0A carry in is T1D carry out
T1INC	25	rw	**T1 Carry Input Selection** 0 T1A carry in is T1D carry out 1 T1A carry in is T0D carry out
T01IN0	[29:28]	rw	**T0 and T1 Global Input CNT0 Selection** 00 Timer T2A overflow/underflow OUV_T2A 01 Positive edge of IN0 10 Negative edge of IN0 11 Both edges of IN0

Field	Bits	Type	Description
T01IN1	[31:30]	rw	**T0 and T1 Global Input CNT1 Selection** 00 Timer T2A overflow/underflow OUV_T2B 01 Positive edge of IN1 10 Negative edge of IN1 11 Both edges of IN1
0	[27:26]	r	**Reserved;** read as 0; should be written with 0.

28.2.1.2 Timer T0/T1 Output, Trigger, and Service Request Selection Register

The T01OTS register performs the selections for the output, service request, and trigger signals of the individual parts of both Timers T0 and T1.

T01OTS
Timer T0 and T1 Output, Trigger and Service Request Selection Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
SOUT00	[1:0]	rw	**T0 Output 0 Source Selection** encoding see **Table 28-3**
SOUT01	[3:2]	rw	**T0 Output 1 Source Selection** encoding see **Table 28-3**
STRG00	[5:4]	rw	**T0 Trigger Output 0 Source Selection** encoding see **Table 28-3**
STRG01	[7:6]	rw	**T0 Trigger Output 1 Source Selection** encoding see **Table 28-3**
SSR00	[9:8]	rw	**T0 Service Request 0 Source Selection** encoding see **Table 28-3**
SSR01	[11:10]	rw	**T0 Service Request 1 Source Selection** encoding see **Table 28-3**
SOUT10	[17:16]	rw	**T1 Output 0 Source Selection** encoding see **Table 28-3**
SOUT11	[19:18]	rw	**T1 Output 1 Source Selection** encoding see **Table 28-3**
STRG10	[21:20]	rw	**T1 Trigger Output 0 Source Selection** encoding see **Table 28-3**

Field	Bits	Type	Description
STRG11	[23:22]	rw	**T1 Trigger Output 1 Source Selection** encoding see **Table 28-3**
SSR10	[25:24]	rw	**T1 Service Request 0 Source Selection** encoding see **Table 28-3**
SSR11	[27:26]	rw	**T1 Service Request 1 Source Selection** encoding see **Table 28-3**
0	[15:12] [31:28]	r	**Reserved;** read as 0; writing to these bit positions has no effect.

Table 28-3 T0/T1 Overflow Source Selection (x, y = 0, 1)

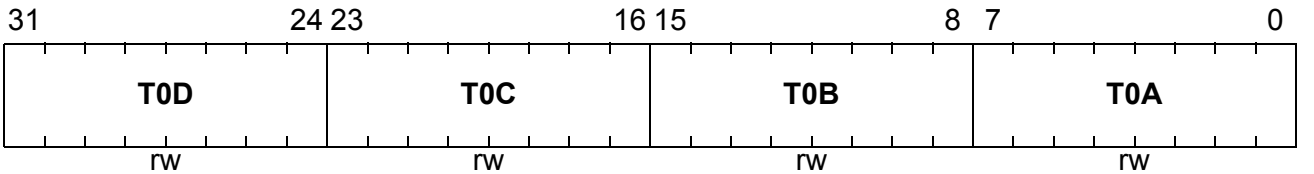
Service Request Selection SSRxy	Trigger Output Selection STRGxy	Output Source Selection SOUTxy	Selected Overflow Signal
00	00	00	TxA overflow
01	01	01	TxB overflow
10	10	10	TxC overflow
11	11	11	TxD overflow

28.2.1.3 Timer T0 and T1 Count and Reload Registers

Timer T0 Count Register T0DCBA (T0D, T0C, T0B, T0A)

This register provides read/write access to all four parts of Timer T0.

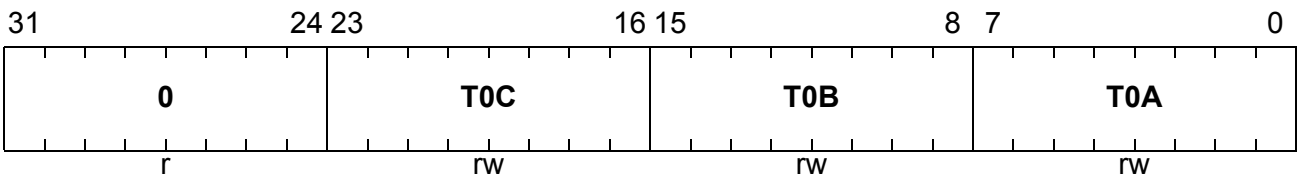
T0DCBA
Timer T0 Count Register (T0D, T0C, T0B, T0A) **Reset Value: 0000 0000$_H$**



Timer T0 Count Register T0CBA (T0C, T0B, T0A)

This register provides read/write access to the lower three parts of Timer T0. The upper byte is always read as 0; writes to it have no effect and are not stored. This register needs to be used if parts A, B, and C of Timer T0 are configured as a 24-bit timer. Part D of Timer T0 will not be affected when writing to this register.

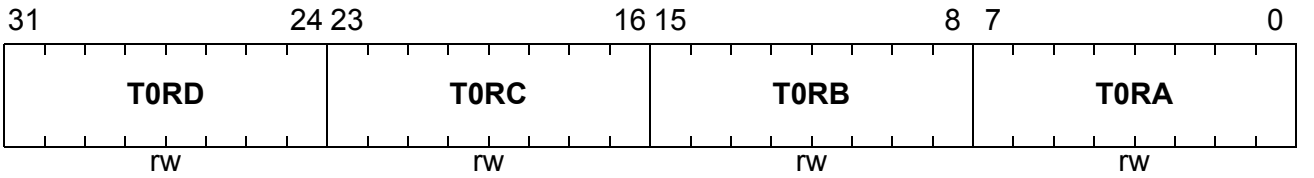
T0CBA
Timer T0 Count Register (T0C, T0B, T0A) **Reset Value: 0000 0000$_H$**



Timer T0 Reload Register T0RDCBA (T0RD, T0RC, T0RB, T0RA)

This register provides read/write access to all four parts of the reload register of Timer T0.

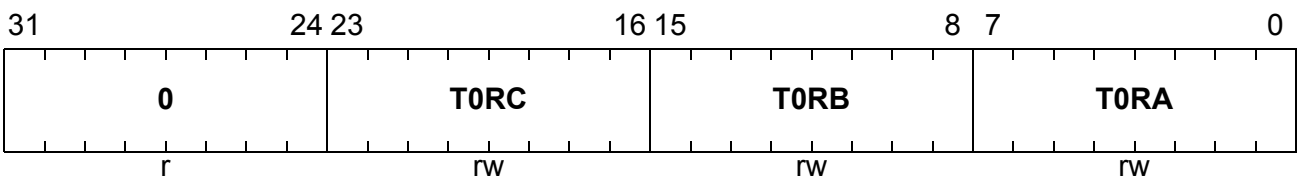
T0RDCBA
Timer T0 Reload Register (T0RD, T0RC, T0RB, T0RA) **Reset Value: 0000 0000$_H$**



T0RCBA, Timer T0 Reload Register (T0RC, T0RB, T0RA)

This register provides read/write access to the lower three parts of the reload register of Timer T0. The upper byte is always read as 0; writes to it have no effect and are not stored. This reload register needs to be used if parts A, B, and C of Timer T0 are configured as a 24-bit timer. Part D of the reload register will not be affected when writing to this register.

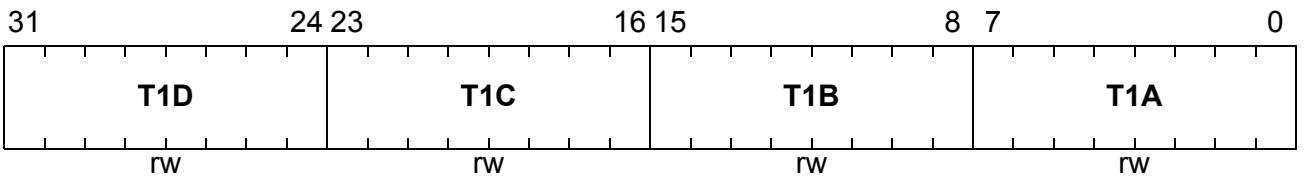
T0RCBA
Timer T0 Reload Register (T0RC, T0RB, T0RA) **Reset Value: 0000 0000$_H$**



Timer T1 Count Register T1DCBA (T1D, T1C, T1B, T1A)

This register provides read/write access to all four parts of Timer T1.

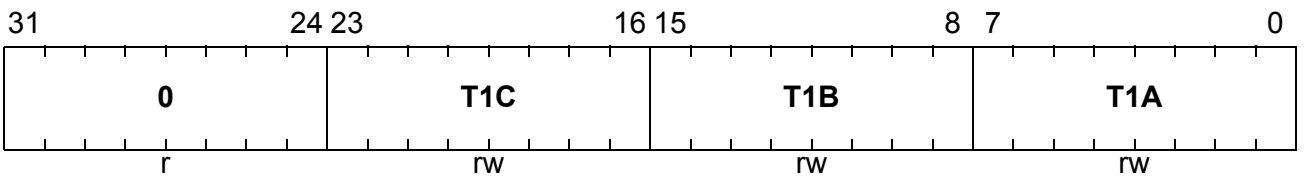
T1DCBA
Timer T1 Count Register (T1D, T1C, T1B, T1A) **Reset Value: 0000 0000$_H$**



Timer T1 Count Register T1CBA (T1C, T1B, T1A)

This register provides read/write access to the lower three parts of Timer T1. The upper byte is always read as 0; writes to it have no effect and are not stored. This register needs to be used if parts A, B, and C of Timer T1 are configured as a 24-bit timer. Part D of Timer T1 will not be affected when writing to this register.

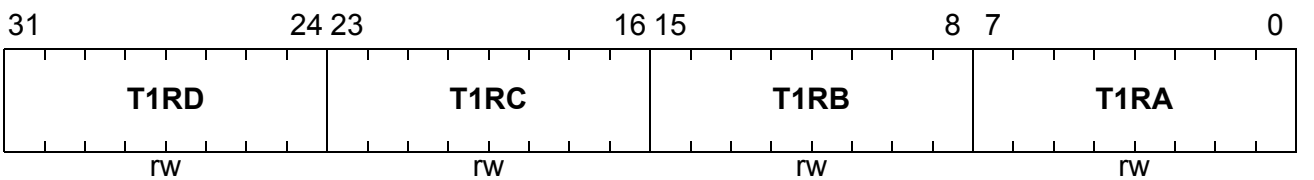
T1CBA
Timer T1 Count Register (T1C, T1B, T1A) **Reset Value: 0000 0000$_H$**



Timer T1 Reload Register T1RDCBA (T1RD, T1RC, T1RB, T1RA)

This register provides read/write access to all four parts of the reload register of Timer T1.

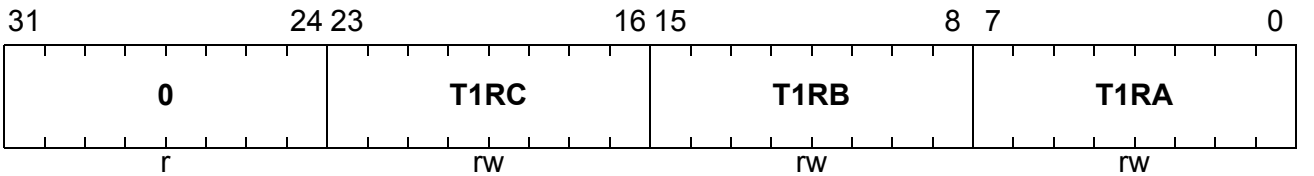
T1RDCBA
Timer T1 Reload Register (T1RD, T1RC, T1RB, T1RA) **Reset Value: 0000 0000$_H$**



Timer T1 Reload Register T1RCBA (T1RC, T1RB, T1RA)

This register provides read/write access to the lower three parts of the reload register of Timer T1. The upper byte is always read as 0; writes to it have no effect and are not stored. This reload register needs to be used if parts A, B, and C of Timer T1 are configured as a 24-bit timer. Part D of the reload register will not be affected when writing to this register.

T1RCBA
Timer T1 Reload Register (T1RC, T1RB, T1RA) **Reset Value: 0000 0000$_H$**



28.2.2 Timer T2 Registers

This section describes the Timer T2 registers.

28.2.2.1 Input Control Registers

Three registers select the input line and the triggering edge for a specific function. The first register, T2AIS, selects the inputs for either Timer T2 in 32-bit mode or Timer T2A in Split Mode. Register T2BIS does the same for Timer T2B in Split Mode. The third register, T2ES, provides the means to select which edge of the selected external signal causes a trigger of the associated function.

Most of these input signals can be used to generate a service request, independent of whether they are used to trigger Timer T2 functions or not.

Timer T2/T2A External Input Selection Register T2AIS

The T2AIS register selects which of the eight external inputs or trigger events from Timer T0/T1 is to be used for the various input functions for Timer T2A. It controls the input selection for Timer T2A in Split Mode and for the entire Timer T2 in 32-bit mode.

T2AIS
Timer T2/T2A External Input Selection Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					T2AIRC1			0	T2AIRC0			0	T2AICLR		
r					rw			r	rw			r	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	T2AIUD			0	T2AISTP			0	T2AISTR			0	T2AICNT		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
T2AICNT	[2:0]	rw	**Timer T2A External Count Input Selection** encoding see **Table 28-4**
T2AISTR	[6:4]	rw	**Timer T2A External Start Input Selection** encoding see **Table 28-4**
T2AISTP	[10:8]	rw	**Timer T2A External Stop Input Selection** encoding see **Table 28-4**
T2AIUD	[14:12]	rw	**Timer T2A External Up/Down Input Selection** encoding see **Table 28-4**

Field	Bits	Type	Description
T2AICLR	[18:16]	rw	**Timer T2A External Clear Input Selection** encoding see **Table 28-4**
T2AIRC0	[22:20]	rw	**Timer T2A External Reload/Capture 0 Input** encoding see **Table 28-4**
T2AIRC1	[26:24]	rw	**Timer T2A External Reload/Capture 1 Input** encoding see **Table 28-4**
0	3, 7, 11,15, 19, 23, [31:27]	r	**Reserved;** read as 0; writing to these bit positions has no effect.

Table 28-4 T2 Input Source Selection (x, y = 0, 1)

Value	Selected External Input	In Parallel Selected Input for T2AICNT, T2AIRC1, and T2AIRC0; and T2BICNT, T2BIRC1, and T2BIRC0
000	Input IN0	T0/T1 Trigger Input Signal TRG00
001	Input IN1	T0/T1 Trigger Input Signal TRG01
010	Input IN2	T0/T1 Trigger Input Signal TRG10
011	Input IN3	T0/T1 Trigger Input Signal TRG11
100	Input IN4	T0/T1 Trigger Input Signal TRG00
101	Input IN5	T0/T1 Trigger Input Signal TRG01
110	Input IN6	T0/T1 Trigger Input Signal TRG10
111	Input IN7	T0/T1 Trigger Input Signal TRG11

*Note: Selection between the input lines and TRGxy is done via the edge selection control (register T2ES, encoding see **Table 28-5**).*

Timer T2B External Input Selection Register T2BIS

The T2BIS register selects which of the external pins or trigger events from Timer T0/T1 is to be used for the various input functions for Timer T2B. This register is used only to select the inputs for Timer T2B in Split Mode; it is inactive in 32-bit mode. The selection is the same as for Timer T2A.

T2BIS
Timer T2B External Input Selection Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					T2BIRC1			0	T2BIRC0			0	T2BICLR		
r					rw			r	rw			r	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	T2BIUD			0	T2BISTP			0	T2BISTR			0	T2BICNT		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
T2BICNT	[2:0]	rw	**Timer T2B External Count Input Selection** encoding see **Table 28-4**
T2BISTR	[6:4]	rw	**Timer T2B External Start Input Selection** encoding see **Table 28-4**
T2BISTP	[10:8]	rw	**Timer T2B External Stop Input Selection** encoding see **Table 28-4**
T2BIUD	[14:12]	rw	**Timer T2B External Up/Down Input Selection** encoding see **Table 28-4**
T2BICLR	[18:16]	rw	**Timer T2B External Clear Input Selection** encoding see **Table 28-4**
T2BIRC0	[22:20]	rw	**Timer T2B External Reload/Capture 0 Input** encoding see **Table 28-4**
T2BIRC1	[26:24]	rw	**Timer T2B External Reload/Capture 1 Input** encoding see **Table 28-4**
0	3, 7, 11,15, 19, 23, [31:27]	r	**Reserved;** read as 0; writing to these bit positions has no effect.

Timer T2 External Input Edge Selection Register T2ES

This register selects the active edge of the external pin input for both Timer T2A and Timer T2B. **Table 28-5** lists the truth table for the edge selection bit fields.

T2ES
Timer 2 External Input Edge Selection Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		T2BERC1		T2BERC0		T2BECLR		T2BEUD		T2BESTP		T2BESTR		T2BECNT	
r		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		T2AERC1		T2AERC0		T2AECLR		T2AEUD		T2AESTP		T2AESTR		T2AECNT	
r		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
T2AECNT	[1:0]	rw	**Timer T2A External Count Input Active Edge Selection** (encoding see **Table 28-5**)
T2AESTR	[3:2]	rw	**Timer T2A External Start Input Active Edge Selection** (encoding see **Table 28-5**)
T2AESTP	[5:4]	rw	**Timer T2A External Stop Input Active Edge Selection** (encoding see **Table 28-5**)
T2AEUD	[7:6]	rw	**Timer T2A External Up/Down Input Active Edge Selection** (encoding see **Table 28-5**)
T2AECLR	[9:8]	rw	**Timer T2A External Clear Input Active Edge Selection** (encoding see **Table 28-5**)
T2AERC0	[11:10]	rw	**Timer T2A External Reload/Capture 0 Input Active Edge Selection** (encoding see **Table 28-5**)
T2AERC1	[13:12]	rw	**Timer T2A External Reload/Capture 1 Input Active Edge Selection** (encoding see **Table 28-5**)
T2BECNT	[17:16]	rw	**Timer T2B External Count Input Active Edge Selection** (encoding see **Table 28-5**)
T2BESTR	[19:18]	rw	**Timer T2B External Start Input Active Edge Selection** (encoding see **Table 28-5**)
T2BESTP	[21:20]	rw	**Timer T2B External Stop Input Active Edge Selection** (encoding see **Table 28-5**)

Field	Bits	Type	Description
T2BEUD	[23:22]	rw	**Timer T2B External Up/Down Input Active Edge Selection** (encoding see **Table 28-5**)
T2BECLR	[25:24]	rw	**Timer T2B External Clear Input Active Edge Selection** (encoding see **Table 28-5**)
T2BERC0	[27:26]	rw	**Timer T2B External Reload/Capture 0 Input Active Edge Selection** (encoding see **Table 28-5**)
T2BERC1	[29:28]	rw	**Timer T2B External Reload/Capture 1 Input Active Edge Selection** (encoding see **Table 28-5**)
0	[15:14], [31:30]	r	**Reserved;** read as 0; writing to these bit positions has no effect.

Table 28-5 T2 Input Source Active Edge Selection

Value	Selected Active Edge	Selected Active Input for T2AECNT, T2AERC1, T2AIRC0; and T2BECNT, T2BERC1, and T2BERC0
00	None	Input is connected to T0/T1 Trigger Input Signal TRGxy as selected in T2xIS
01	Positive edge	–
10	Negative edge	–
11	Both edges	–

28.2.2.2 Mode Control and Status Register

Two registers control the mode of operation for the timer and the reload/capture registers. They also provide status information. The first register, T2CON, controls the operation of the timer itself and holds the status information, while the second register, T2RCCON, controls the operation of the two reload/capture registers.

The T2CON register controls the operating mode of Timer T2. The control bits and functions are the same for Timer T2A and Timer T2B.

T2CON
Timer 2 Mode Control and Status Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			T2BDIR	0			T2BCOS	T2BCOV		T2BCCLR		T2BCDIR		T2BCSRC	
r			rw	r			rw	rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T2SPLIT	0		T2ADIR	0			T2ACOS	T2ACOV		T2ACCLR		T2ACDIR		T2ACSRC	
rw	r		rw	r			rw	rw		rw		rw		rw	

Field	Bits	Type	Description
T2ACSRC	[1:0]	rw	**Timer T2A Count Input Source Control** encoding see **Table 28-9**
T2ACDIR	[3:2]	rw	**Timer T2A Direction Control** encoding see **Table 28-8**
T2ACCLR	[5:4]	rw	**Timer T2A Clear Control** encoding see **Table 28-7**
T2ACOV	[7:6]	rw	**Timer T2A Overflow/Underflow Generation Control** (encoding see **Table 28-6**)
T2ACOS	8	rw	**Timer T2A One-Shot Control** 0 T2A continues to run after overflow or underflow 1 T2A stops after the first overflow or underflow
T2ADIR	12	rw	**Timer T2A Direction Status Flag** 0 T2A Direction is up-counting 1 T2A Direction is down-counting

Field	Bits	Type	Description
T2SPLIT	15	rw	**Timer T2 Split Control** 0 Timer T2 operates as one 32-bit timer, controlled via T2A controls 1 Timer T2 operates as two independent 16-bit timers T2A and T2B
T2BCSRC	[17:16]	rw	**Timer T2B Count Input Source Control** encoding see **Table 28-9**
T2BCDIR	[19:18]	rw	**Timer T2B Direction Control** encoding see **Table 28-8**
T2BCCLR	[21:20]	rw	**Timer T2B Clear Control** encoding see **Table 28-7**
T2BCOV	[23:22]	rw	**Timer T2B Overflow/Underflow Generation Control** (encoding see **Table 28-6**)
T2BCOS	24	rw	**Timer T2B One-Shot Control** 0 T2B continues to run after overflow or underflow 1 T2B stops after the first overflow or underflow
T2BDIR	28	rw	**Timer T2B Direction Status Flag** 0 T2B direction is up-counting 1 T2B direction is down-counting
0	[11:9], [14:13], [27:25], [31:29]	r	**Reserved;** read as 0; writing to these bit positions has no effect.

Table 28-6 T2 Overflow/Underflow Generation Control

T2BCOV T2ACOV	Selected Function
00	Overflow is generated for FF..FF$_H$ → 00..00$_H$; Underflow is generated for 00..00$_H$ → FF..FF$_H$
01	Overflow is generated for FF..F**E**$_H$ → FF..FF$_H$; underflow is generated for 00..00$_H$ → FF..FF$_H$
10	Overflow is generated for FF..FF$_H$ → 00..00$_H$; underflow is generated for 00..0**1**$_H$ → 00..00$_H$
11	Overflow is generated for FF..F**E**$_H$ → FF..FF$_H$; Underflow is generated for 00..0**1**$_H$ → 00..00$_H$

Table 28-7 T2 Clear Control

T2BCCLR T2ACCLR[1)]	Selected Function
00	Clear timer to $00..00_H$ on external event (Clear_B/Clear_A)
01	Clear timer on capture 0 event (CP0_T2B/CP0_T2A)
10	Clear timer on capture 1 event (CP1_T2B/CP1_T2A)
11	Reserved. Do not use this combination.

1) In Clear-on-Capture mode, the timer contents are first captured, then the timer is cleared.

Table 28-8 T2 Direction Control

T2BCDIR T2ACDIR	Selected Function[1)]
00	Count direction is count up (software controlled).
01	Count direction is count down (software controlled).
10[2)]	Count direction controlled through external signal (UpDown_B / UpDown_A). Count up if external signal is 1, else count down.
11[2)]	Count direction controlled through external signal (UpDown_B / UpDown_A). Count down if external signal is 1, else count up.

1) If Quadrature Counting is selected, the count direction is controlled through the relation of the two signals Count_A/B and Up/Down A/B; the bit fields T2ACDIR/T2BCDIR have no effect in this case.

2) The last two options have an extra line going from the input selection to the direction control representing the state of the input (not shown in the diagrams). The edge selection has no effect on the direction control; however, it can be used to generate a service request (UpDown_A only).

Table 28-9 T2 Count Input Source Control

T2BCSRC T2ACSRC	Selected Function
00	Count input source is the module clock f_{GPTU}.
01	Count input source is external count input Count_x.
10	Quadrature Counter Mode. Count input sources are the two inputs Count_x and UpDown_x.
11	Reserved. Do not use this combination.

28.2.2.3 Timer T0/T1/T2 Run Control Register

The run control bits of the individual parts of timers T0, T1, and T2 are all contained in register T012RUN. This register allows synchronous starting or stopping of several or all timers with one instruction.

T012RUN
Timer T0, T1, and T2 Run Control Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	T2B CLR R	T2B SET R	T2B RUN	0	T2A CLR R	T2A SET R	T2A RUN	T1D RUN	T1C RUN	T1B RUN	T1A RUN	T0D RUN	T0C RUN	T0B RUN	T0A RUN
r	w	w	rh	r	w	w	rh	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
T0ARUN	0	rw	**Timer T0A Run Control** 0 Stop T0A 1 Start T0A
T0BRUN	1	rw	**Timer T0B Run Control** 0 Stop T0B 1 Start T0B
T0CRUN	2	rw	**Timer T0C Run Control** 0 Stop T0C 1 Start T0C
T0DRUN	3	rw	**Timer T0D Run Control** 0 Stop T0D 1 Start T0D
T1ARUN	4	rw	**Timer T1A Run Control** 0 Stop T1A 1 Start T1A
T1BRUN	5	rw	**Timer T1B Run Control** 0 Stop T1B 1 Start T1B

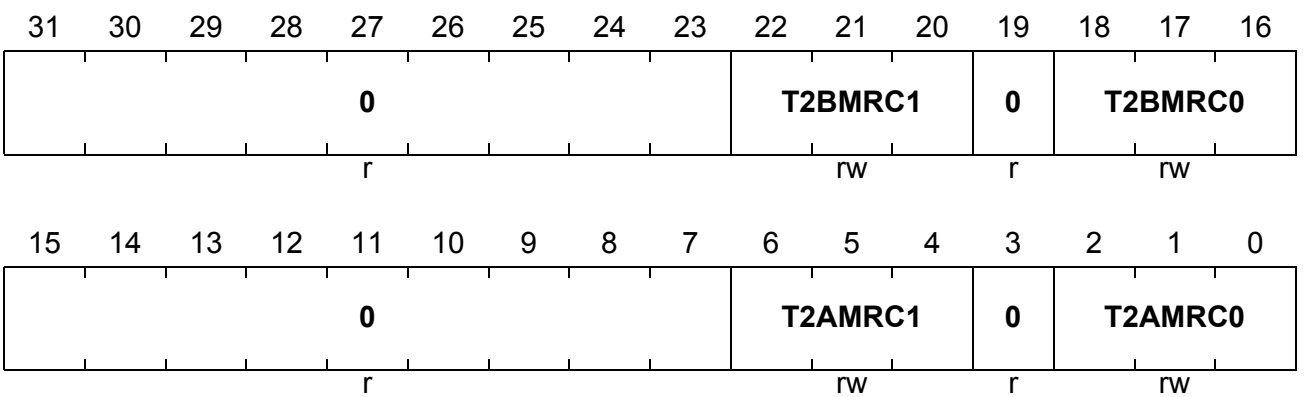
Field	Bits	Type	Description
T1CRUN	6	rw	**Timer T1C Run Control** 0 Stop T1C 1 Start T1C
T1DRUN	7	rw	**Timer T1D Run Control** 0 Stop T1D 1 Start T1D
T2ARUN	8	rh	**Timer T2A Run Status Flag** 0 T2A is stopped 1 T2A is running This bit indicates the running/stopped status of Timer T2A. This status bit can be directly set or reset by hardware depending on the selections and external events causing a start or a stop of the timer. It can only be affected by software through the set and clear bits T2ASETR and T2ACLRR, respectively. Writing directly to this bit via software has no effect.
T2ASETR	9	w	**Timer T2A Run Set Bit** Writing a 1 to this bit causes the run bit T2ARUN to be set to 1, thus starting Timer T2A. Possible hardware modifications of T2ARUN that occurred during read-modify-write instructions (for example, bit set, bit clear instructions) are lost; the software modification has priority. The value written to T2ASETR is not stored. Writing a 0 to this bit has no effect. This bit always returns 0 when read. If both T2ASETR and T2ACLRR are set, T2ARUN is not affected.
T2ACLRR	10	w	**Timer T2A Run Clear Bit** Writing a 1 to this bit causes the run bit T2ARUN to be cleared, thus stopping timer T2A. Possible hardware modifications of T2ARUN that occurred during read-modify-write instructions (for example, bit set, bit clear instructions) are lost; the software modification has priority. The value written to T2ACLRR is not stored. Writing a 0 to this bit has no effect. This bit always returns 0 when read. If both T2ASETR and T2ACLRR are set, T2ARUN is not affected.

Field	Bits	Type	Description
T2BRUN	12	rh	**Timer T2B Run Status Flag** 0 T2B is stopped 1 T2B is running More details see description for T2ARUN.
T2BSETR	13	w	**Timer T2B Run Set Bit** More details see description for T2ASETR.
T2BCLRR	14	w	**Timer T2B Run Clear Bit** More details see description for T2ACLRR.
0	11, [31:15]	r	**Reserved;** read as 0; writing to these bit positions has no effect.

28.2.2.4 T2 Reload/Capture Mode Control Register

This register selects the reload/capture mode operation for the reload/capture registers T2ARC0, T2ARC1, T2BRC0, and T2BRC1.

T2RCCON
Timer 2 Reload/Capture Mode Control Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
T2AMRC0	[2:0]	rw	**Timer T2A Reload/Capture 0 Mode Control** encoding see **Table 28-10**
T2AMRC1	[6:4]	rw	**Timer T2A Reload/Capture 1 Mode Control** encoding see **Table 28-10**
T2BMRC0	[18:16]	rw	**Timer T2B Reload/Capture 0 Mode Control** encoding see **Table 28-10**
T2BMRC1	[22:20]	rw	**Timer T2B Reload/Capture 1 Mode Control** encoding see **Table 28-10**

Field	Bits	Type	Description
0	3, 19, [15:7], [31:23]	r	**Reserved;** read as 0; writing to these bit positions has no effect.

Table 28-10 T2 Capture/Reload Mode Selection

T2AMRCx T2BMRCx	Selected Operation for T2ARC0/T2BRC0	Selected Operation for T2ARC1/T2BRC1
000	Disabled	
001	Reserved. Do not use this combination.	
010	Reserved. Do not use this combination.	
011	Capture on external event	
100	Reload on overflow or underflow	
101	Reload on external event	
110	Reload on overflow only	Reload on underflow only
111	Reload on external event if count direction is up (if T2ADIR/T2BDIR = 0)	Reload on external event if count direction is down (T2ADIR/T2BDIR = 1)

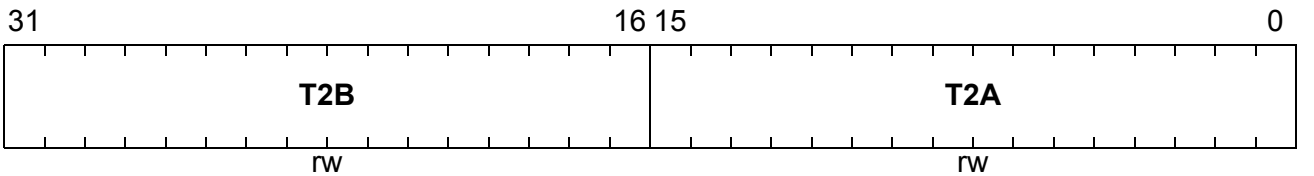
Note: If a capture event for one register and a reload event for the other register occur at the same time, the timer contents are captured first; then, the timer is reloaded. If both reload/capture registers are set up for reload and the trigger events occur at the same time for both, only the reload from the higher numbered register (T2ARC1/T2BRC1) is performed.

28.2.2.5 Timer T2 Count and Reload/Capture Registers

Timer T2 Count Register

Register T2 holds the actual count value of Timer T2. In Split Mode, the lower half-word of this register represents the contents of Timer T2A, while the upper half-word represents the contents of Timer T2B. Proper load/store instructions must be used depending on whether the timer is operated in full 32-bit or in Split Mode.

T2
Timer T2 Count Register **Reset Value: 0000 0000$_H$**

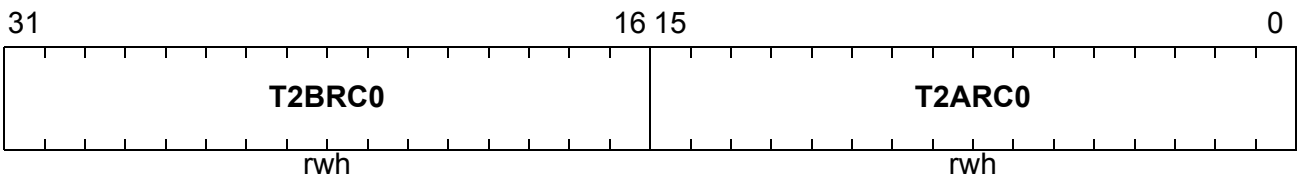


Field	Bits	Type	Description
T2A	[15:0]	rwh	**T2A Contents** (in Split Mode)
T2B	[31:16]	rwh	**T2B Contents** (in Split Mode)

Timer T2 Reload/Capture Registers

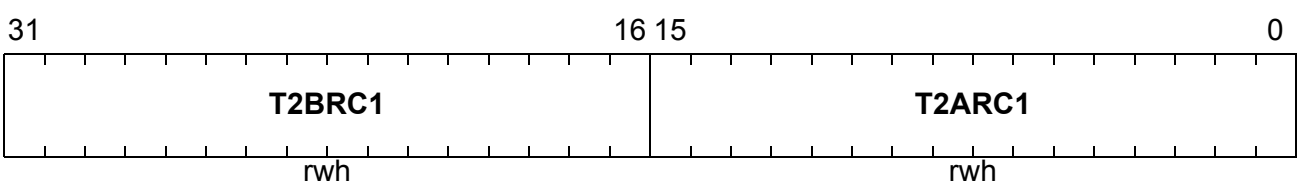
The two reload/capture values for Timer T2 are held in registers T2RC0 and T2RC1, respectively. In Split Mode, the lower half-word of these registers represent the respective Timer T2A reload/capture values (T2ARC0, T2ARC1), while the upper half-word is used for the Timer T2B reload/capture values (T2BRC0, T2BRC1). The same access mechanisms apply here as for the timer count register Timer T2.

T2RC0
Timer T2 Reload/Capture Register 0 **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
T2ARC0	[15.0]	rwh	**T2A Reload/Capture Value** (in Split Mode) In Capture Mode, the register contents are also affected by hardware.
T2BRC0	[31:16]	rwh	**T2B Reload/Capture Value** (in Split Mode) In Capture Mode, the register contents are also affected by hardware.

T2RC1
Timer T2 Reload/Capture Register 1 **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
T2ARC1	[15:0]	rwh	**T2A Reload/Capture Value** (in Split Mode) In Capture Mode, the register contents are also affected by hardware.
T2BRC1	[31:16]	rwh	**T2B Reload/Capture Value** (in Split Mode) In Capture Mode, the register contents are also affected by hardware.

28.2.3 Global Control Registers

The OSEL register selects the output source function for the output state bits.

OSEL
Output Source Selection Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**SO7**		**0**		**SO6**		**0**		**SO5**		**0**		**SO4**	
r		rw		r		rw		r		rw		r		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**SO3**		**0**		**SO2**		**0**		**SO1**		**0**		**SO0**	
r		rw		r		rw		r		rw		r		rw	

Field	Bits	Type	Description
SO0	[2:0]	rw	**Output 0 Source Selection** see **Table 28-11** for encoding
SO1	[6:4]	rw	**Output 1 Source Selection** encoding see **Table 28-11**
SO2	[10:8]	rw	**Output 2 Source Selection** encoding see **Table 28-11**
SO3	[14:12]	rw	**Output 3 Source Selection** encoding see **Table 28-11**
SO4	[18:16]	rw	**Output 4 Source Selection** encoding see **Table 28-11**
SO5	[22:20]	rw	**Output 5 Source Selection** encoding see **Table 28-11**
SO6	[26:24]	rw	**Output 6 Source Selection** encoding see **Table 28-11**
SO7	[30:28]	rw	**Output 7 Source Selection** encoding see **Table 28-11**
0	3, 7, 11, 15, 19, 23, 27, 31	r	**Reserved;** read as 0; writing to these bit positions has no effect.

Table 28-11 T2 Output Signal Source Selection

Value	Selected Source
000	OUT00
001	OUT01
010	OUT10
011	OUT11
100	OUV_T2A
101	OUV_T2B
110	Reserved. Do not use these combinations.
111	

Each output has an output state bit, OUTx. These bits toggle each time a trigger signal occurs. The state of these bits can be made available at the respective output pins through the alternate function selections at these pins. The output state bits and the enable bits are contained in the output control register OUT.

The output state bits can also be modified by software. Individual set and clear bits are provided for each of the output state bits. Software can update a state bit via these separate bits only.

OUT
Output Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								SETO7	SETO6	SETO5	SETO4	SETO3	SETO2	SETO1	SETO0
r								w	w	w	w	w	w	w	w

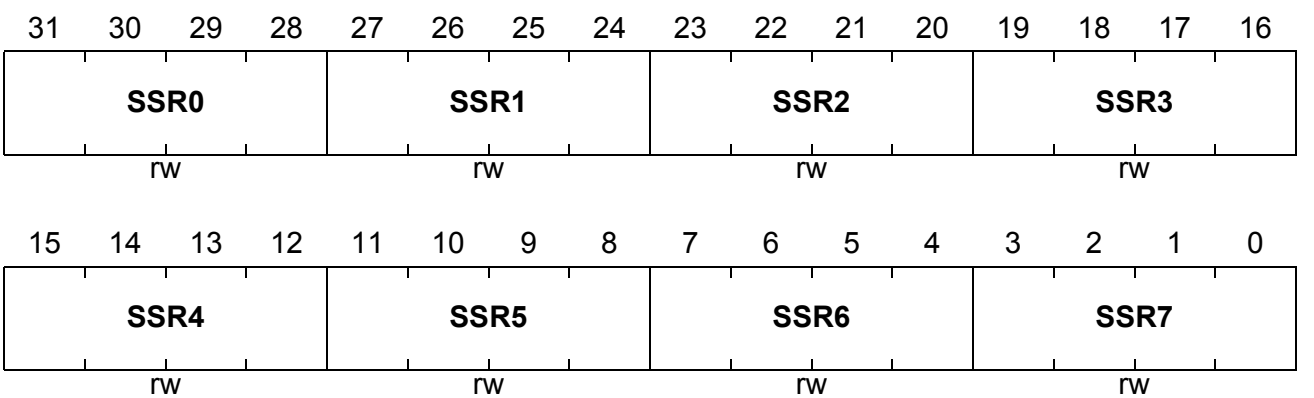
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRO7	CLRO6	CLRO5	CLRO4	CLRO3	CLRO2	CLRO1	CLRO0	OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
w	w	w	w	w	w	w	w	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
OUTx (x = 0-7)	[7:0]	rh	**Output x Status Bit** This status bit can be directly set or reset by the associated trigger event. It can be set or reset only by software via writing a 1 to either bit SETOx or bit CLROx, respectively. Writing directly to this bit via software has no effect.

Field	Bits	Type	Description
CLROx (x = 0-7)	[15:8]	w	**Output x Clear Bit** Writing a 1 to this bit causes the output bit OUTx to be cleared. Possible hardware modifications of OUTx that occurred during read-modify-write instructions (for example, bit set, bit clear instructions) are lost; the software modification has priority. The value written to CLROx is not stored. Writing a 0 to this bit has no effect. This bit always returns 0 when read. If both SETOx and CLROx are set, OUTx is not affected.
SETOx (x = 0-7)	[23:16]	w	**Output x Set Bit** Writing a 1 to this bit causes the output bit OUTx to be set to 1. Possible hardware modifications of OUTx that occurred during read-modify-write instructions (for example, bit set, bit clear instructions) are lost; the software modification has priority. The value written to SETOx is not stored. Writing a 0 to this bit has no effect. This bit always returns 0 when read. If both SETOx and CLROx are set, OUTx is not affected.
0	[31:24]	r	**Reserved;** read as 0; writing to these bit positions has no effect.

This SRSEL register selects which of the various events in the Timer T0, T1, and T2 blocks generate one of the eight service requests.

SRSEL
Service Request Source Selection Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
SSR7	[3:0]	rw	**Service Request Node 7 Source Selection** encoding see **Table 28-12**
SSR6	[7:4]	rw	**Service Request Node 6 Source Selection** encoding see **Table 28-12**
SSR5	[11:8]	rw	**Service Request Node 5 Source Selection** encoding see **Table 28-12**
SSR4	[15:12]	rw	**Service Request Node 4 Source Selection** encoding see **Table 28-12**
SSR3	[19:16]	rw	**Service Request Node 3 Source Selection** encoding see **Table 28-12**
SSR2	[23:20]	rw	**Service Request Node 2 Source Selection** encoding see **Table 28-12**
SSR1	[27:24]	rw	**Service Request Node 1 Source Selection** encoding see **Table 28-12**
SSR0	[31:28]	rw	**Service Request Node 0 Source Selection** encoding see **Table 28-12**

Table 28-12 T2 Service Request Source Selection

Value	Selected Source
0000	Start_A
0001	Stop_A
0010	UpDown_A
0011	Clear_A
0100	RLCP0_A
0101	RLCP1_A
0110	OUV_T2A
0111	OUV_T2B
1000	Start_B
1001	Stop_B
1010	RLCP0_B
1011	RLCP1_B
1100	SR00
1101	SR01
1110	SR10
1111	SR11

28.3 GPTU Module Implementation

This section describes the GPTU Module interfaces with the clock control, port connections, interrupt control, and address decoding.

28.3.1 Interfaces of the GPTU Modules

Figure 28-16 shows the TC1130 specific implementation details and interconnections of the GPTU Module. The GPTU Module has eight I/O lines located at Port 0. Further, the GPTU Module is supplied by a separate clock control, interrupt control and address decoding logic.

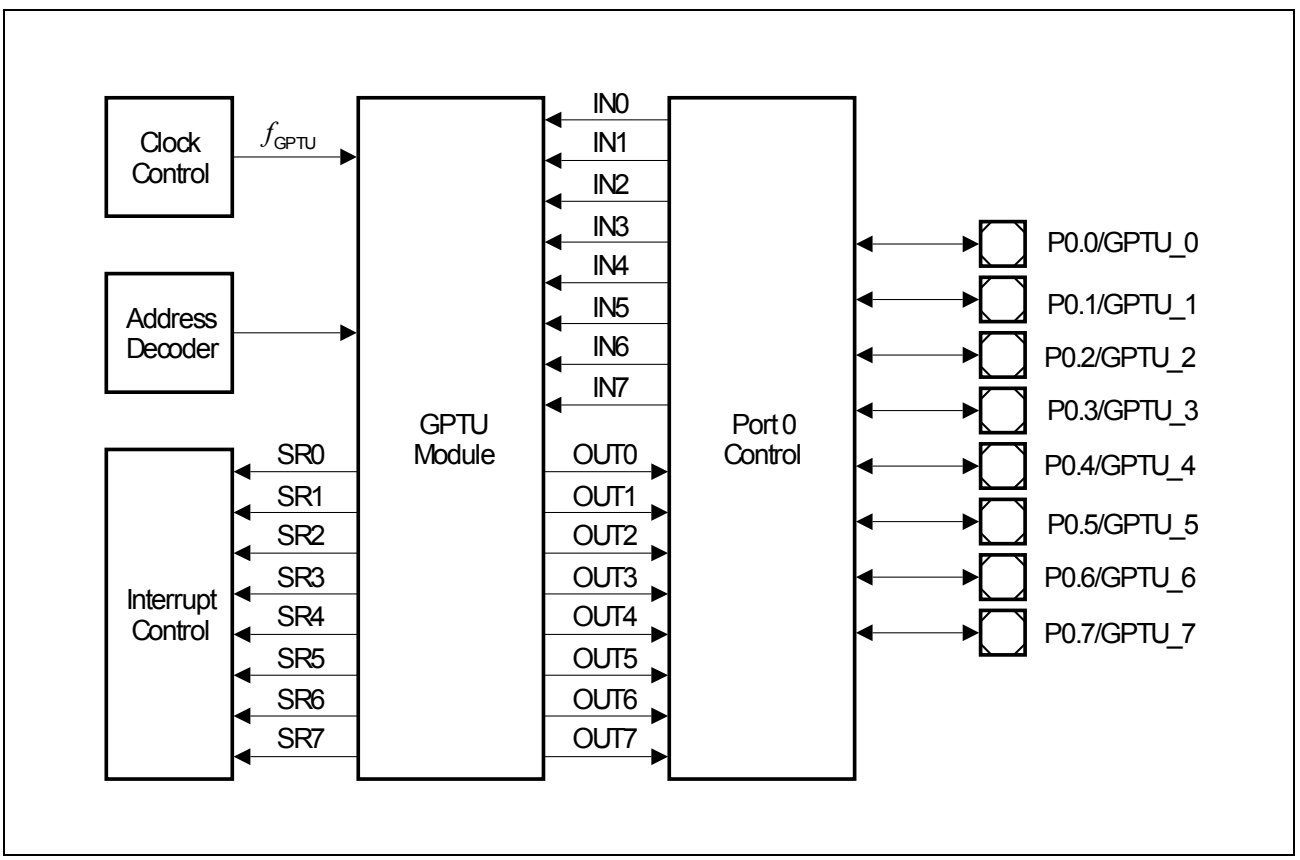


Figure 28-16 GPTU Module Implementation and Interconnections

28.3.2 GPTU Module Related External Registers

Figure 28-17 summarizes the module related external registers which are required for GPTU programming (see also **Figure 28-15** for the module kernel specific registers).

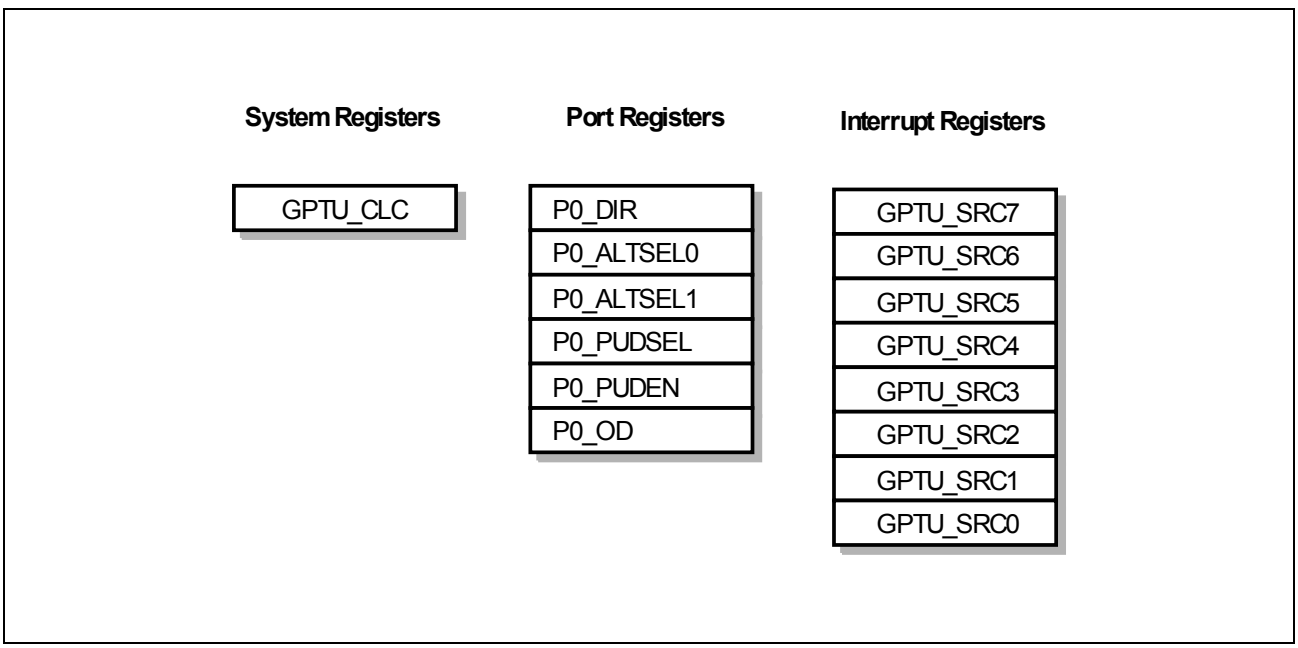
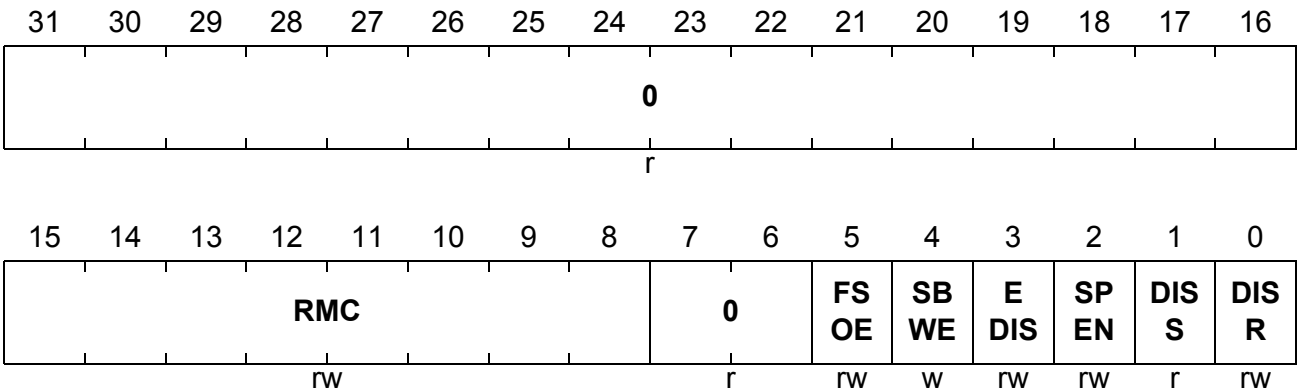


Figure 28-17 GPTU Implementation Specific Special Function Registers

28.3.2.1 Clock Control Registers

The clock control register allows the programmer to adapt the functionality and power consumption of a GPTU Module to the requirements of the application. The diagram below shows the clock control register functionality as implemented for the GPTU Module.

GPTU_CLC
GPTU Clock Control Register **Reset Value: 0000 0002$_H$**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the module.
SPEN	2	rw	**Module Suspend Enable for OCDS** Used for enabling the suspend mode.
EDIS	3	rw	**External Request Disable** Used for controlling the external clock disable request.
SBWE	4	w	**Module Suspend Bit Write Enable for OCDS** Defines whether SPEN and FSOE are write protected.
FSOE	5	rw	**Fast Switch Off Enable** Used for fast clock switch off in OCDS suspend mode.
RMC	[15:8]	rw	**8-Bit Clock Divider Value in RUN Mode**
0	[7:6], [31:16]	r	**Reserved;** read as 0; should be written with 0.

28.3.2.2 Port Control

The interconnections between the GPTU modules and the port I/O lines are controlled in the port logics. The following port control operations selections must be executed:

- Input/output direction selection (DIR registers)
- Alternate function selection (ALTSEL0 and ALTSEL1 registers)
- Input/Output driver characteristic control (PUDSEL, PUDEN and OD registers)

Input/Output Function Selection

The port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as pull-up/down devices, port direction (input/output), open-drain, and alternate output selections. The I/O lines for the GPTU modules are controlled by the port input/output control registers of Port0.

Table 28-13 shows how bits and bit fields must be programmed for the required I/O functionality of the GPTU I/O lines.

Table 28-13 GPTU I/O Line Selection and Setup

Module	Port Lines	Input/Output Control Register Bits	I/O
GPTU	P0.0/GPTU_0	P0_DIR.P0 = 0_B	Input
		P0_DIR.P0 = 1_B	Output
		P0_ALTSEL0.P0 = 1_B	
		P0_ALTSEL1.P0 = 0_B	
	P0.1/GPTU_1	P0_DIR.P1 = 0_B	Input
		P0_DIR.P1 = 1_B	Output
		P0_ALTSEL0.P1 = 1_B	
		P0_ALTSEL1.P1 = 0_B	
	P0.2/GPTU_2	P0_DIR.P2 = 0_B	Input
		P0_DIR.P2 = 1_B	Output
		P0_ALTSEL0.P2 = 1_B	
		P0_ALTSEL1.P2 = 0_B	
	P0.3/GPTU_3	P0_DIR.P3 = 0_B	Input
		P0_DIR.P3 = 1_B	Output
		P0_ALTSEL0.P3 = 1_B	
		P0_ALTSEL1.P3 = 0_B	
	P0.4/GPTU_4	P0_DIR.P4 = 0_B	Input
		P0_DIR.P4 = 1_B	Output
		P0_ALTSEL0.P4 = 1_B	
		P0_ALTSEL1.P4 = 0_B	
	P0.5/GPTU_5	P0_DIR.P5 = 0_B	Input
		P0_DIR.P5 = 1_B	Output
		P0_ALTSEL0.P5 = 1_B	
		P0_ALTSEL1.P5 = 0_B	
	P0.6/GPTU_6	P0_DIR.P6 = 0_B	Input
		P0_DIR.P6 = 1_B	Output
		P0_ALTSEL0.P6 = 1_B	
		P0_ALTSEL1.P6 = 0_B	

Table 28-13 GPTU I/O Line Selection and Setup (cont'd)

Module	Port Lines	Input/Output Control Register Bits	I/O
	P0.7/GPTU_7	P0_DIR.P7 = 0_B	Input
		P0_DIR.P7 = 1_B	Output
		P0_ALTSEL0.P7 = 1_B	
		P0_ALTSEL1.P7 = 0_B	

P0_DIR
Port 0 Direction Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-7)**	n	rw	**Port 0 Pin 0 - 7 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for GPTU I/O port control.

P0_ALTSELn (n = 0, 1)
Port 0 Alternate Select Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 28-14 Function of the Bits P0_ALTSEL0.Pn and P0_ALTSEL1.Pn (n = 0-7)[1)]

P0_ALTSEL0.Pn	P0_ALTSEL1.Pn	Function
1	0	Alternate Select 1

1) Shaded bits and bit field are don't care for GPTU I/O port control.

The GPTU ports also offer the possibility to configure the following output characteristics:

- Push/pull (optional pull-up/pull-down)
- Open drain with internal pull-up
- Open drain with external pull-up

P0_PUDSEL
Port 0 Pull-Up/Pull-Down Select Register **Reset Value: 0000 FFFF$_H$**

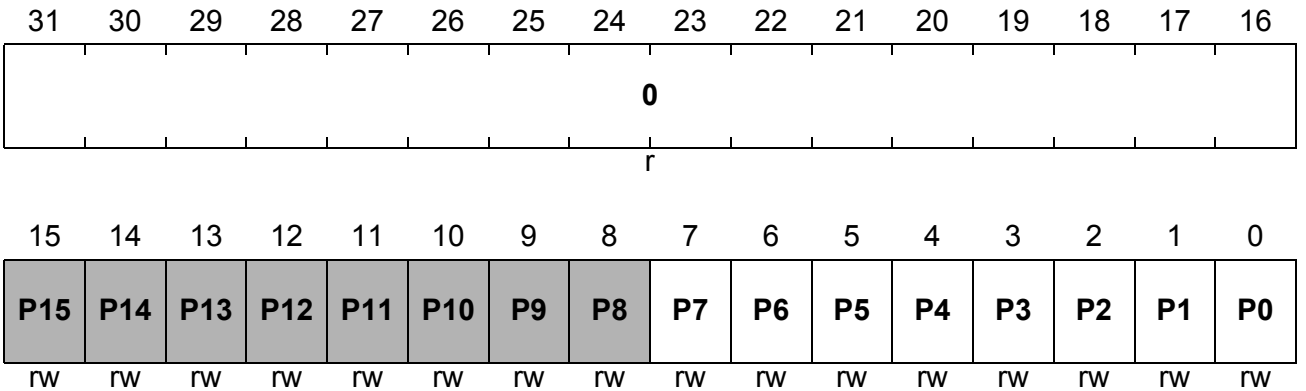
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-7)**	n	rw	**Pull-Up/Pull-Down Select Port 0 Bit n**[1)] 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for GPTU I/O port control.

P0_PUDEN
Port 0 Pull-Up/Pull-Down Enable Register **Reset Value: 0000 FFFF$_H$**



Field	Bits	Type	Description
Pn **(n = 0-7)**	n	rw	**Pull-Up/Pull-Down Enable at Port 0 Bit n**[1)] 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for GPTU I/O port control.

P0_OD
Port 0 Open Drain Control Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-7)**	n	rw	**Port 0 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for GPTU I/O port control.

28.3.2.3 Interrupt Registers

The eight interrupt outputs SR7 - SR0 of each GPTU Module are controlled by the service request control registers GPTU_SRC7 to GPTU_SRC0.

GPTU_SRC0
GPTU Interrupt Service Request Control Register 0
GPTU_SRC1
GPTU Interrupt Service Request Control Register 1
GPTU_SRC2
GPTU Interrupt Service Request Control Register 2
GPTU_SRC3
GPTU Interrupt Service Request Control Register 3
GPTU_SRC4
GPTU Interrupt Service Request Control Register 4
GPTU_SRC5
GPTU Interrupt Service Request Control Register 5
GPTU_SRC6
GPTU Interrupt Service Request Control Register 6
GPTU_SRC7
GPTU Interrupt Service Request Control Register 7 **Reset Values: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETR	CLRR	SRR	SRE	0	TOS	0		SRPN							
w	w	rh	rw	r	rw	r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number**
TOS	10	rw	**Type of Service Control**
SRE	12	rw	**Service Request Enable**
SRR	13	rh	**Service Request Flag**
CLRR	14	w	**Request Clear Bit**
SETR	15	w	**Request Set Bit**

Field	Bits	Type	Description
0	[9:8], 11, [31:16]	r	**Reserved;** read as 0; should be written with 0.

Note: Further details on interrupt handling and processing are described in the "Interrupt System" chapter of the TC1130 System Units User's Manual.

28.3.3 GPTU Register Address Ranges

In the TC1130, the registers of the GPTU Module is located in the following address ranges:

- Module Base Address = F000 0600_H
- Module End Address = F000 $06FF_H$
- Absolute Register Address = Module Base Address + Offset Address (offset addresses see **Table 28-2**)

Note: The complete and detailed address map of the GPTU module is described in the chapter "Register Overview" of the TC1130 System Units User's Manual.

29 Capture/Compare Unit 6 (CCU6)

This chapter describes the two Capture/Compare Unit 6 Modules CCU60 and CCU61 of the TC1130. It contains the following sections:

- Functional description of a CCU6 kernel (see **Section 29.1**)
- CCU6 kernel register descriptions (see **Section 29.2**)
- TC1130 implementation specific details and registers of the CCU6 modules (port connections and control, interrupt control, address decoding, and clock control, see **Section 29.3**)

*Note: The CCU6 kernel register names described in **Section 29.2** will be referenced in the TC1130 User's Manual by the module name prefix "CCU60_" for the CCU60 interface and by "CCU61_" for the CCU61 interface.*

29.1 CCU6 Kernel Description

29.1.1 Overview

The CCU6 provides two independent timers (T12, T13) that can be used for Pulse Width Modulation (PWM) generation, especially for AC-motor control. Additionally, special control modes for block commutation and multi-phase machines are supported.

The timer T12 can work in capture and/or compare mode for its three channels. The modes can also be combined. The timer T13 can work in compare mode only. The multi-channel control unit generates output patterns which can be modulated by T12 and/or T13. The modulation sources can be selected and combined for the signal modulation.

Timer 12 Features

- Three capture/compare channels, each channel can be used either as capture or as compare channel
- Generation of a three-phase PWM supported (six outputs, individual signals for highside and lowside switches)
- 16-bit resolution, maximum count frequency = peripheral clock
- Dead-time control for each channel to avoid short-circuits in the power stage
- Concurrent update of the required T12/T13 registers
- Center-aligned and edge-aligned PWM can be generated
- Single-shot mode supported
- Many interrupt request sources
- Hysteresis-like control mode

Timer 13 Features

- One independent compare channel with one output
- 16-bit resolution, maximum count frequency = peripheral clock
- Can be synchronized to T12
- Interrupt generation at period-match and compare-match
- Single-shot mode supported

Additional Features

- Block commutation for Brushless DC-drives implemented
- Position detection via Hall-sensor pattern
- Automatic rotational speed measurement for block commutation
- Integrated error handling
- Fast emergency stop without CPU load via external signal ($\overline{\text{CTRAP}}$)
- Control modes for multi-channel AC-drives
- Output levels can be selected and adapted to the power stage

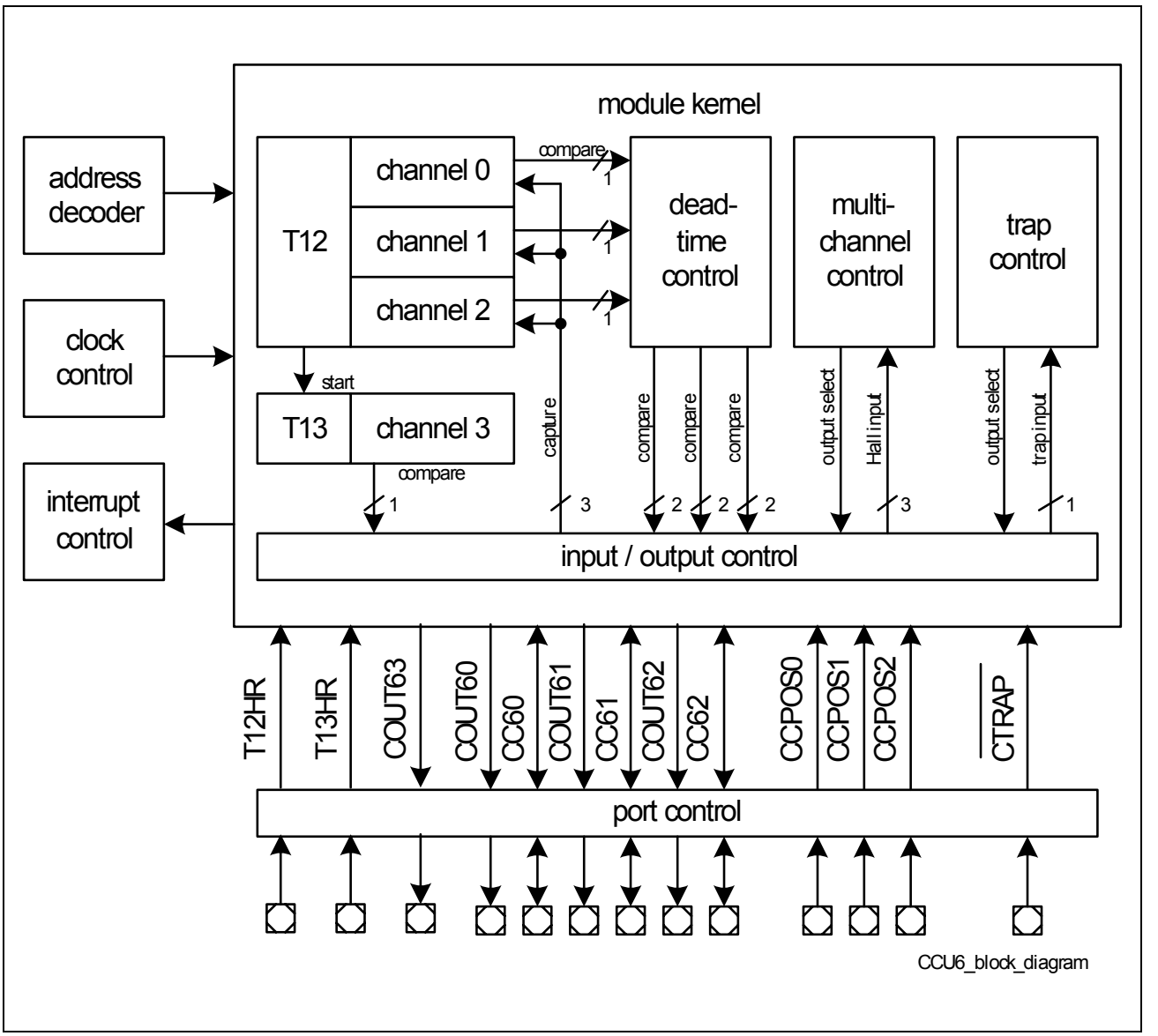


Figure 29-1 CCU6 Block Diagram

29.1.2 Timer T12

29.1.2.1 Overview

The timer T12 is used for capture/compare purposes with three independent channels. The timer T12 is a 16-bit wide counter. Three channel registers (CC60R, CC61R, CC62R), which are built with shadow registers (CC60SR, CC61SR, CC62SR), contain the compare value or the captured timer value. In compare mode, the software writes to the shadow registers and their contents are transferred simultaneously to the actual compare registers during the T12 shadow transfer. In capture mode, the captured value of T12 can be read from the channel registers. The period of the timer T12 is fixed by the period register T12PR, which is also built with a shadow register.

The write access from the CPU targets the corresponding shadow registers, whereas the read access targets the registers actually used (except for the three compare channels, where the actual and the shadow registers can be read).

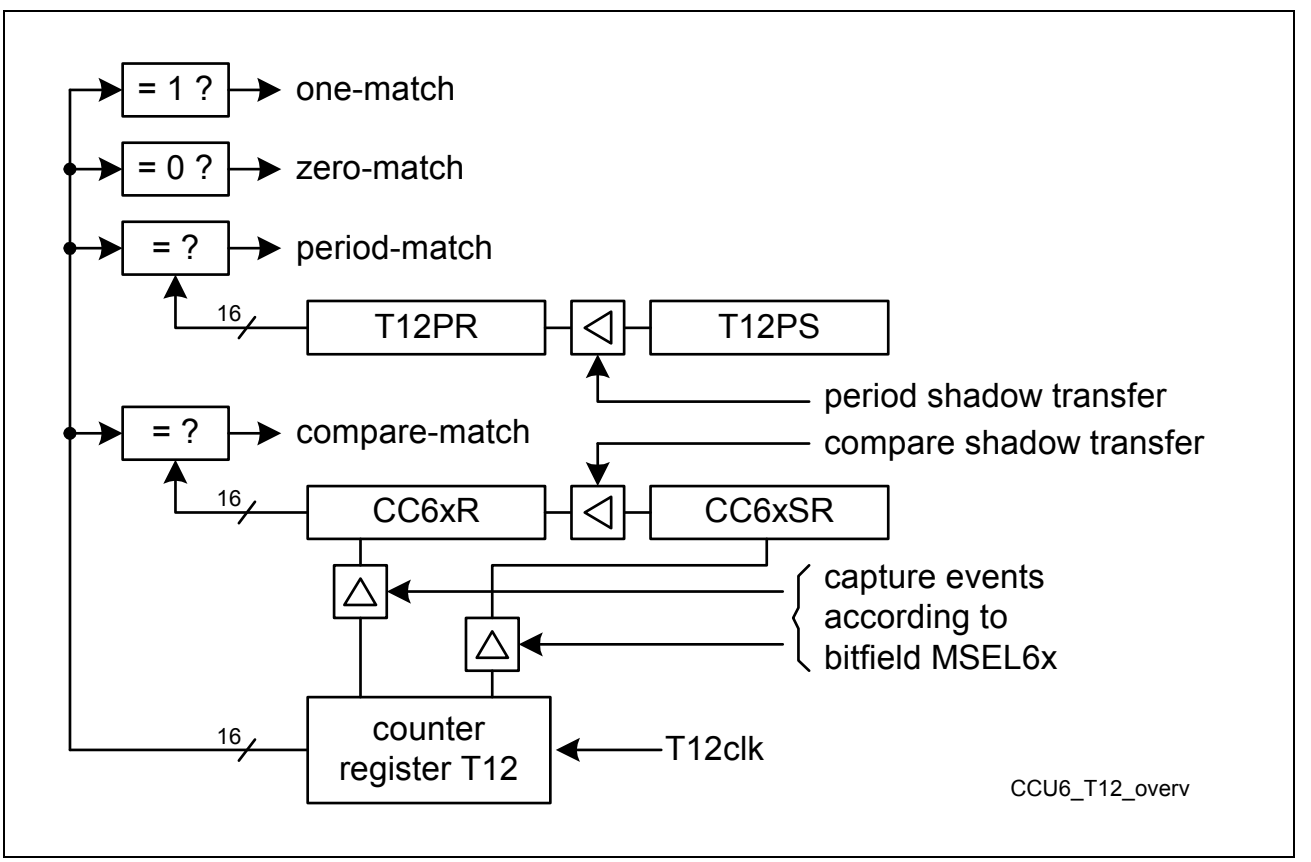


Figure 29-2 T12 Overview

While timer T12 is running, write accesses to register T12 are not taken into account. If the timer T12 is stopped and the dead-time counters are 0, write actions to register T12 are immediately taken into account.

29.1.2.2 Counting Rules

With reference to the T12 input clock, the counting sequence is defined by the following counting rules:

T12 in edge-aligned mode:

- The counter is reset to zero and (if desired) the T12 shadow transfer takes place if a period-match is detected. The counting direction is always upwards.

T12 in center-aligned mode:

- The count direction is set to counting up (CDIR = 0) if a one-match is detected while counting down.
- The count direction is set to counting down (CDIR = 1) if a period-match is detected while counting up.
- The counter counts up while CDIR = 0 and it counts down while CDIR = 1.
- If enabled, shadow transfer takes place:
  – if a period-match is detected while counting up
  – if a one-match is detected while counting down

The timer T12 prescaler is reset while T12 is not running to ensure reproducible timings and delays.

The counting rules lead to the following sequences:

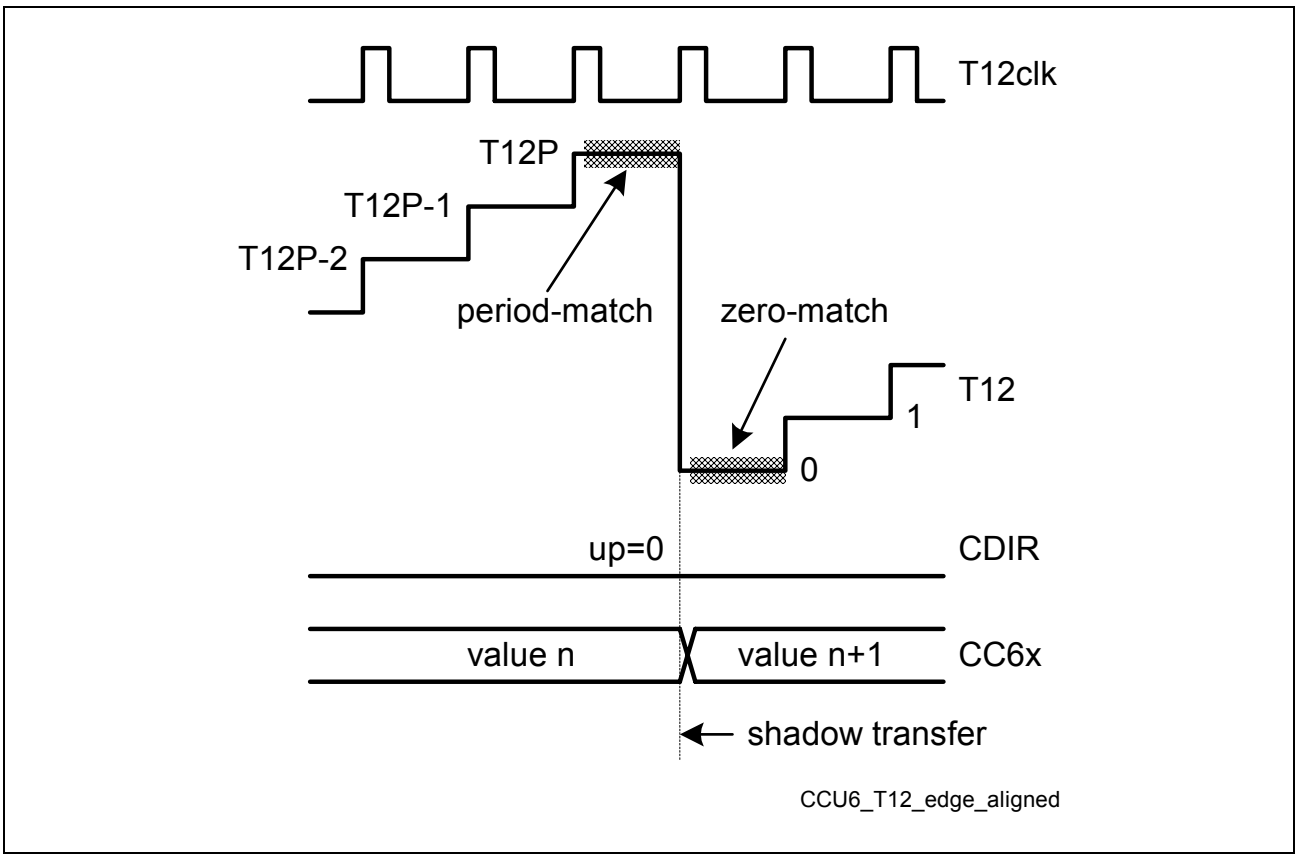


Figure 29-3 T12 in Edge-aligned Mode

In the center-aligned mode (T12 counts up and down), the counting rules lead to the following behavior:

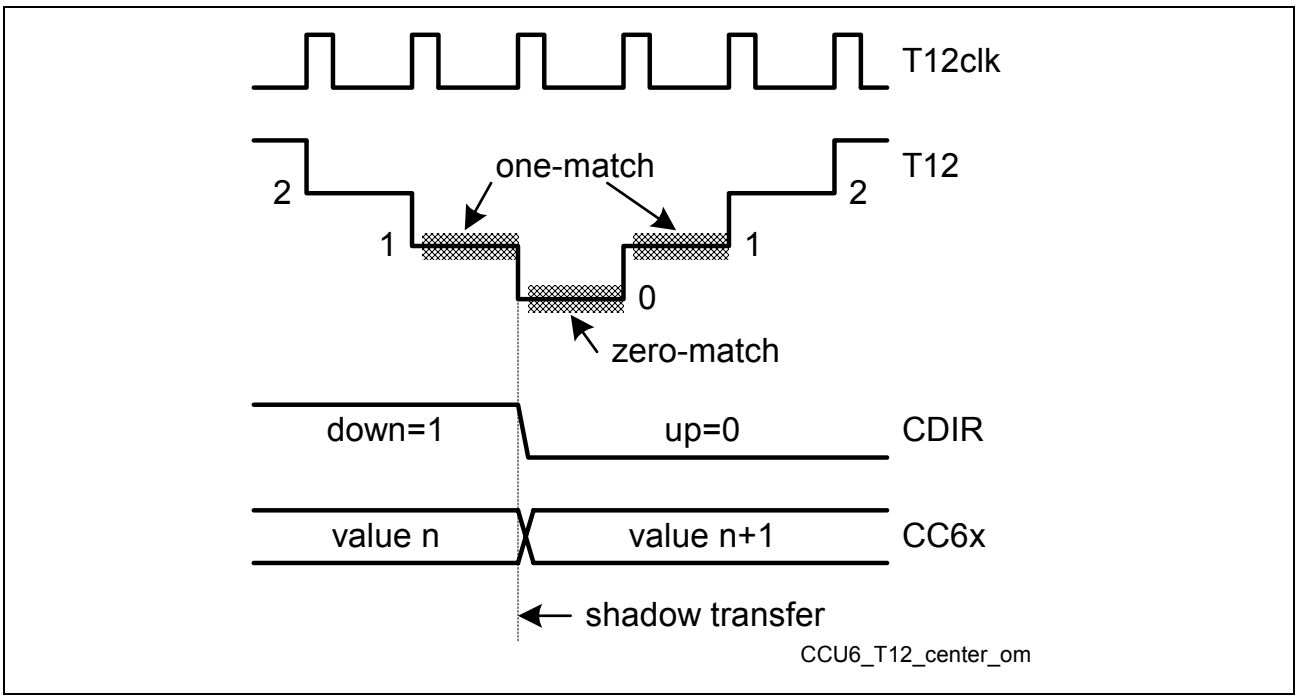


Figure 29-4 T12 in Center-aligned Mode, One-match Detected

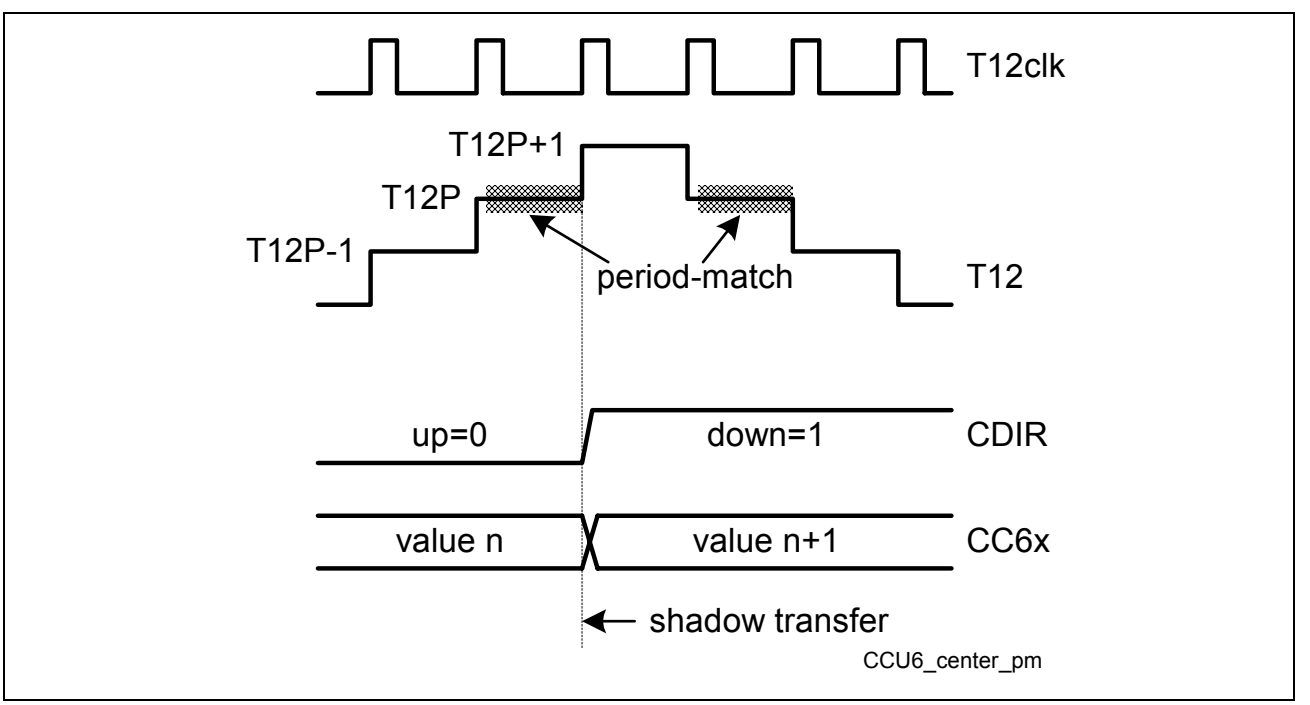


Figure 29-5 T12 in Center-aligned Mode, Period-match Detected

29.1.2.3 Switching Rules

The compare actions take place in parallel for the three compare channels. Depending on the count direction, the compare matches have different meanings. In order to get the PWM information independent from the output levels, two different states have been introduced for the compare actions: the active state and the passive state. Both these states are used to generate the desired PWM as a combination of the states delivered by T13, the trap control unit and the multi-channel control unit. If the active state is interpreted as a 1 and the passive state as a 0, the state information is combined with a logical AND function.

- Active AND active = active
- Active AND passive = passive
- Passive AND passive = passive

The compare states change with the detected compare-matches and are indicated by the CC6xST bits. The compare states of T12 are defined as follows:

- Passive if the counter value is below the compare value
- Active if the counter value is above the compare value

This leads to the following switching rules for the compare states:

- Set to the active state when the counter value reaches the compare value while counting up.
- Reset to the passive state when the counter value reaches the compare value while counting down.
- Reset to the passive state in case of a zero-match without compare-match while counting up.
- Set to the active state in case of a zero-match with a parallel compare-match while counting up.

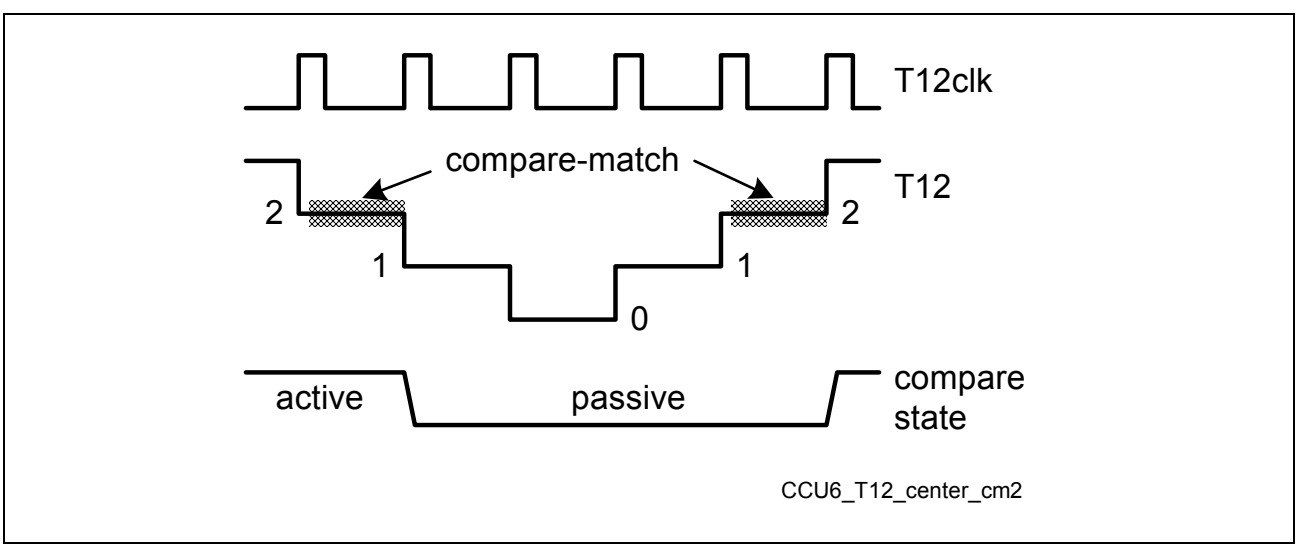


Figure 29-6 Compare States for Compare Value = 2

The switching rules are only taken into account while the timer is running. As a result, write actions to the timer registers while the timer is stopped do not lead to compare actions.

29.1.2.4 Duty Cycle of 0% and 100%

These counting and switching rules ensure a PWM functionality in the full range between 0% and 100% duty cycle (duty cycle = active time/total PWM period). In order to obtain a duty cycle of 0% (compare state never active), a compare value of T12P+1 must be programmed (for both compare modes). A compare value of 0 will lead to a duty cycle of 100% (compare state always active).

29.1.2.5 External Timer Start

The timer run bit T12R can also be set automatically if an event is detected on the external signal T12HR. The event can be a rising edge, a falling edge or any edge; see **Figure 29-7**. If bit field T12RSEL = 00_B, the external timer start feature is disabled and the timer run bit can be only controlled by software.

The set and reset conditions of the timer run bit T12R is kept unchanged due to T12 functionality.

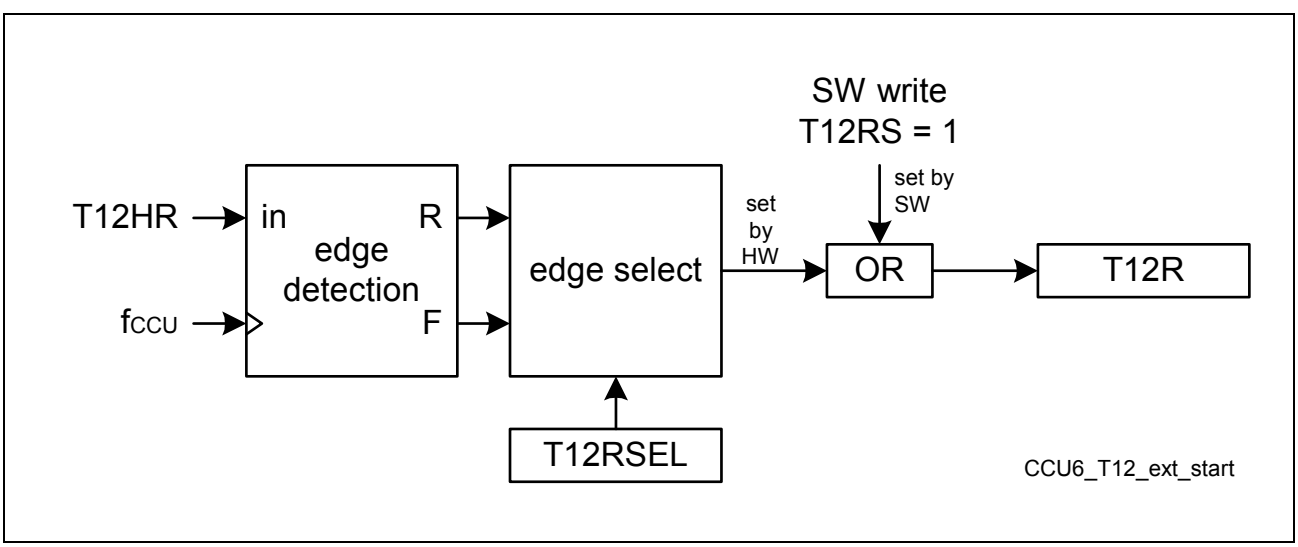


Figure 29-7 External Timer Start

29.1.2.6 Compare Mode of T12

Figure 29-8 shows the setting and resetting of the compare state bit CC6xST. In order to simplify the description, only one out of the three parallel channels is described. The letter “x” in the simplified bit names and signal names indicates that there are more than one channel. The CC6xST bit is the compare state bit in register CMPSTAT, the bit CC6xPS represents passive state select bit.

The timer T12 generates pulses indicating events like compare-matches, period-matches and zero-matches, which are used to set (signal T12_xST_se) and to reset (signal T12_xST_re) the corresponding compare state bit (CC6xST) according to the counting direction.

The timer T12 modulation output lines T12xO (two for each channel) can be selected to be in the active state while the corresponding compare state is 0 (with CC6xPS = 0) or while the corresponding compare state is 1 (with CC6xPS = 1). The bit COUT6xPS has the same effect for the second output of the channel. The example is shown without dead-time.

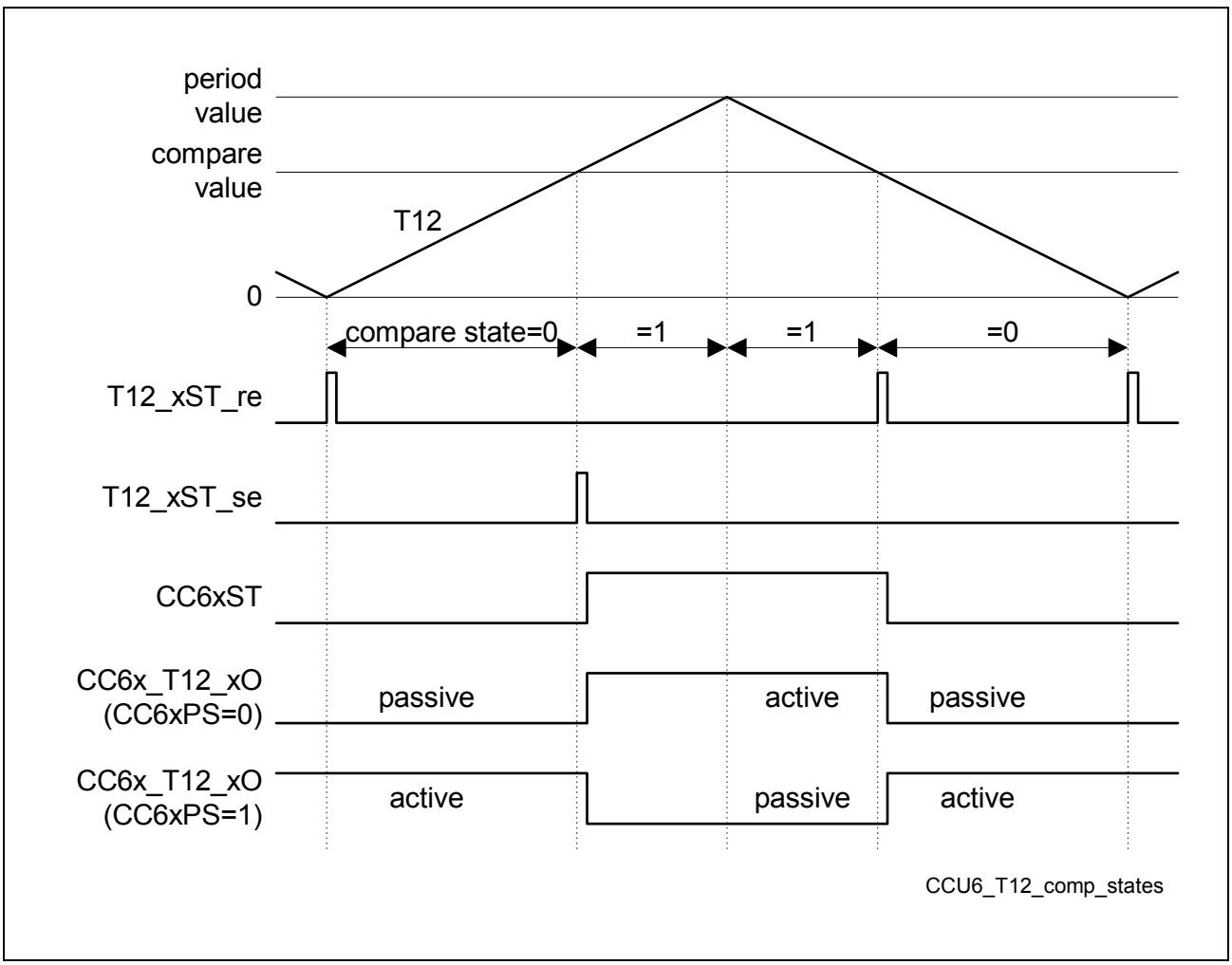


Figure 29-8 Compare States of Timer T12

According to the desired capture/compare mode, the compare state bits must be switched. Therefore, an additional logic (see **Figure 29-9**) selects how and by which event the compare state bits are modified. The mode selection (by bit fields MSEL6x in register T12MSEL) enables the setting and the resetting of the compare state bits due to compare actions of timer T12.

The hardware modification of the compare state bits is only possible while the timer T12 is running. Therefore, the bit T12R is used to enable/disable the modification by hardware.

For the hysteresis-like compare mode (MSEL6x = 1001_B), the setting of the compare state bit is only possible while the corresponding input CCPOSx = 1 (inactive).

If the Hall Sensor mode (MSEL6x = 1000_B) is selected, the compare state bits of the compare channels 1 and 2 are modified by the timer T12 in order to indicate that a programmed time has elapsed.

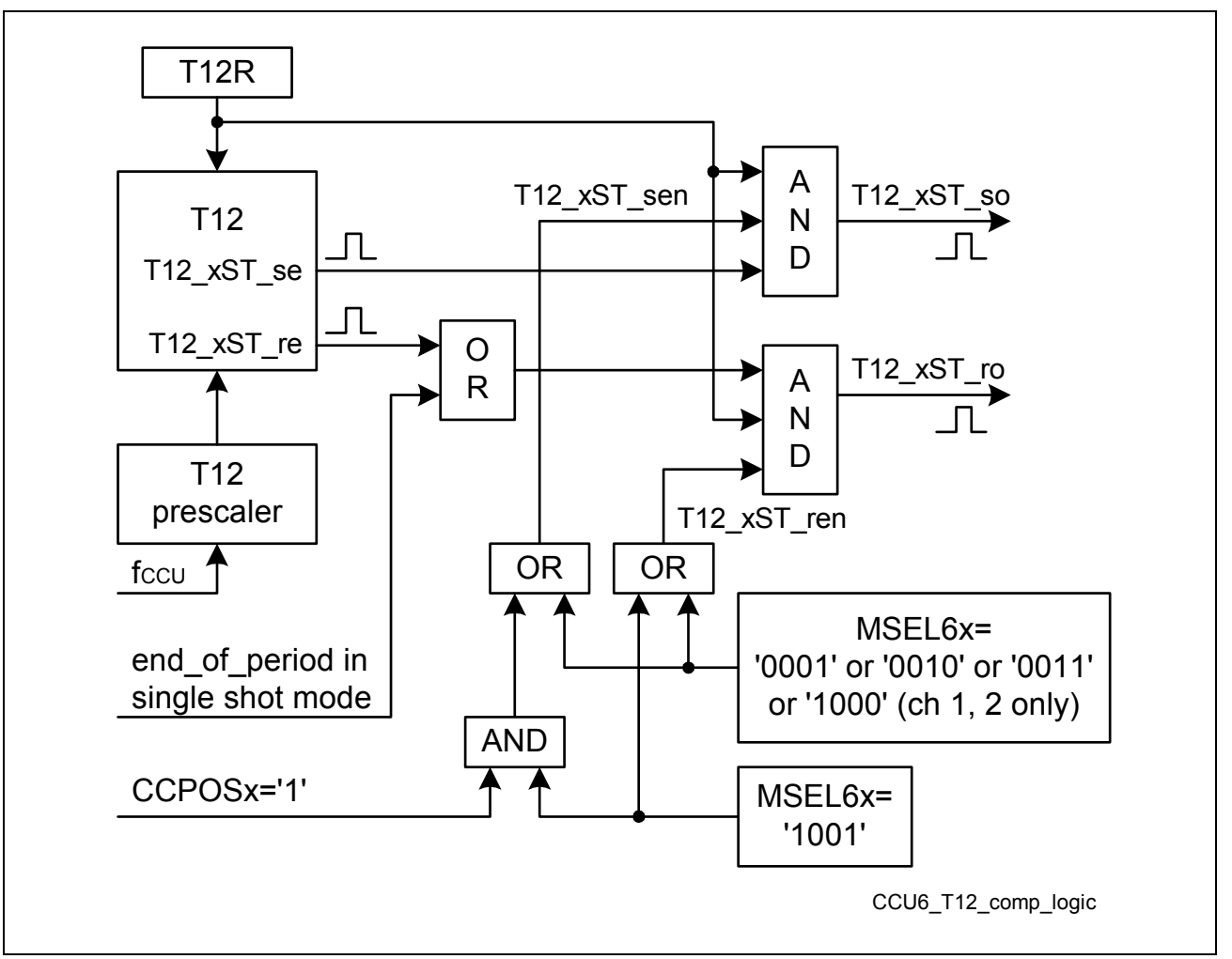


Figure 29-9 T12 Compare Logic

The T12 compare output lines T12_xST_so (to set bit CC6xST) and T12_xST_ro (to reset bit CC6xST) are also used to trigger the corresponding interrupt flags and to

generate interrupts. The signal T12_xST_so indicates the interrupt event for the rising edge (ICC6xR), whereas the signal T12_xST_ro indicates the falling edge event (ICC6xF) in compare mode.

The compare state bits indicate the occurrence of a capture or compare event of the corresponding channel. It can be set (if it is 0) by the following events:

- upon a software set (MCC6xS)
- upon a compare set event (see switching rules) if the T12 runs and if the T12 set event is enabled
- upon a capture set event

The bit CC6xST can be reset (if it is 1) by the following events:

- upon a software reset (MCC6xR)
- upon a compare reset event (see switching rules) if the T12 runs and if the T12 reset event is enabled (including in single shot mode at the end of the T12 period)
- upon a reset event in the hysteresis-like control mode

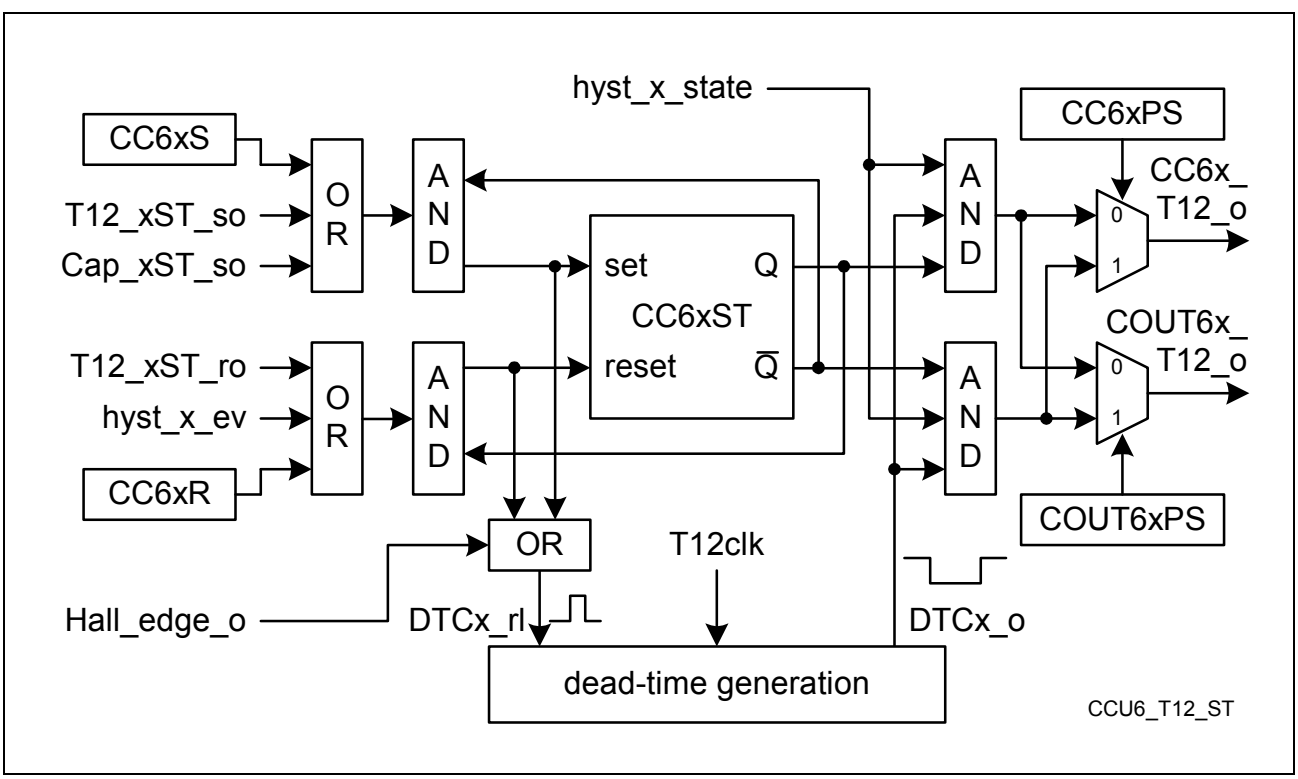


Figure 29-10 T12 Logic for CC6xST Control

The events triggering the set and reset action of the CC6xST bits must be combined; see **Figure 29-10**. The occurrence of the selected capture event (signal Cap_xST_so) or the setting of CC6xS in register CMPMODIF also leads to a set action of bit CC6xST, whereas the negative edge at pin CCPOSx (in hysteresis-like mode, signal hyst_x_ev) or the setting of bit CC6xR leads to reset action.

The set signal is only generated while bit CC6xST is reset, a reset can only take place while the bit is set. This permits the OR-combination of the resulting set and reset signals to one common signal (DTCx_rl) triggering the reload of the dead-time counter. It is only triggered if bit CC6xST is changed, permitting a correct PWM generation with dead-time and the complete duty cycle range of 0% to 100% in edge-aligned and in center-aligned mode.

In the case that the dead-time generation is enabled, the change of bit CC6xST triggers the dead-time unit and a signal DTCx_o is generated. The length of the 0 level of this signal corresponds to the desired dead-time, which is used to delay the rising edge (passive to active edge) of the output signal.

In order to generate independent PWM patterns for the highside and the lowside switches of the power inverter, the interval when a PWM signal should be active can be selected by the bits CC6xPS. They select if the PWM signal is active while the compare state bit is 0 (T12 counter value below the compare value) or while it is 1 (T12 counter value above the compare value).

In **Figure 29-10**, the signals CC6x_T12_o and COUT6x_T12_o are inputs to the modulation control block, where they can be combined with other PWM signals.

29.1.2.7 Switching Examples in Edge-aligned Mode

Figure 29-11 shows two switching examples in edge-aligned mode with duty cycles near to 0% and near to 100%. The compare-, period- or zero-matches lead to modifications of the compare state and the shadow transfer (if requested by STE12 = 1) in the next clock cycle.

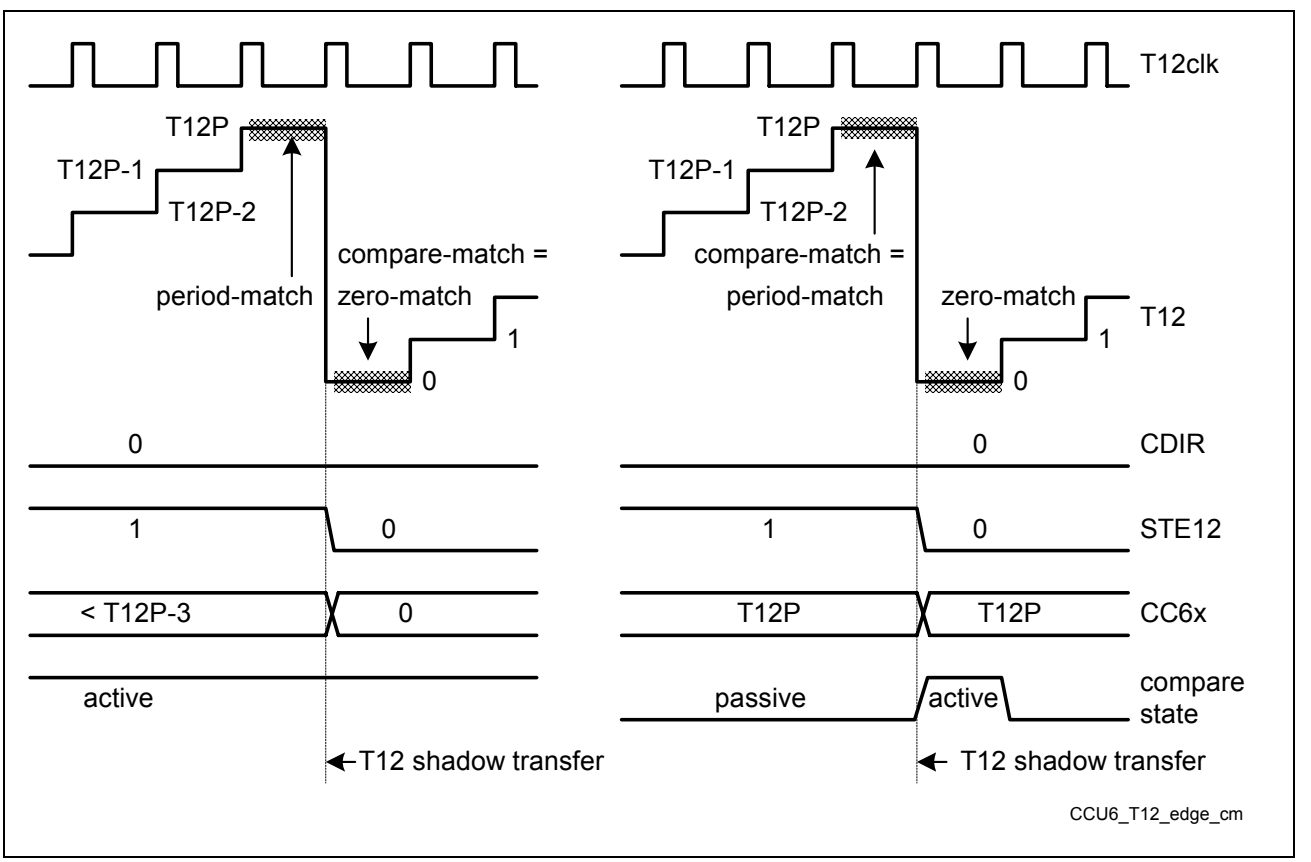


Figure 29-11 Switching Examples in Edge-aligned Mode

29.1.2.8 Switching Examples in Center-aligned Mode

The following figures show examples of the switching of the compare state and the T12 shadow transfer according to the programmed compare values.

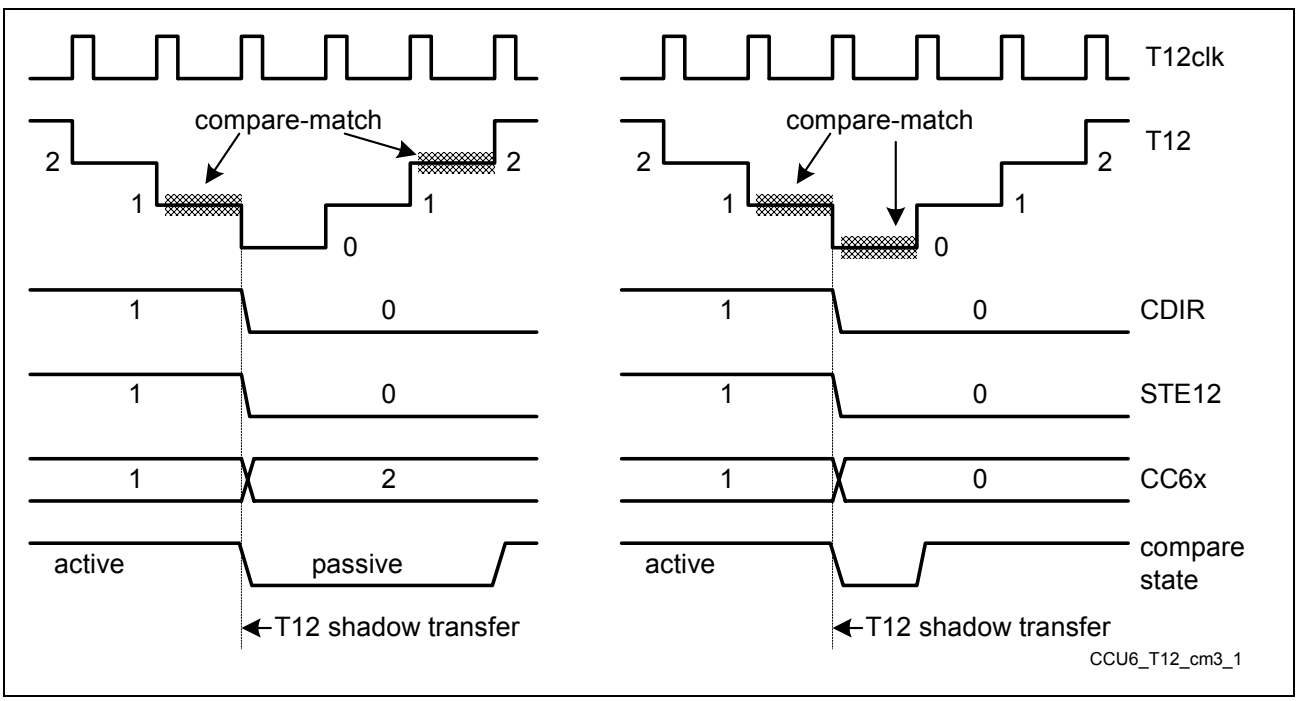


Figure 29-12 Switching Example for Duty Cycles near to 100%

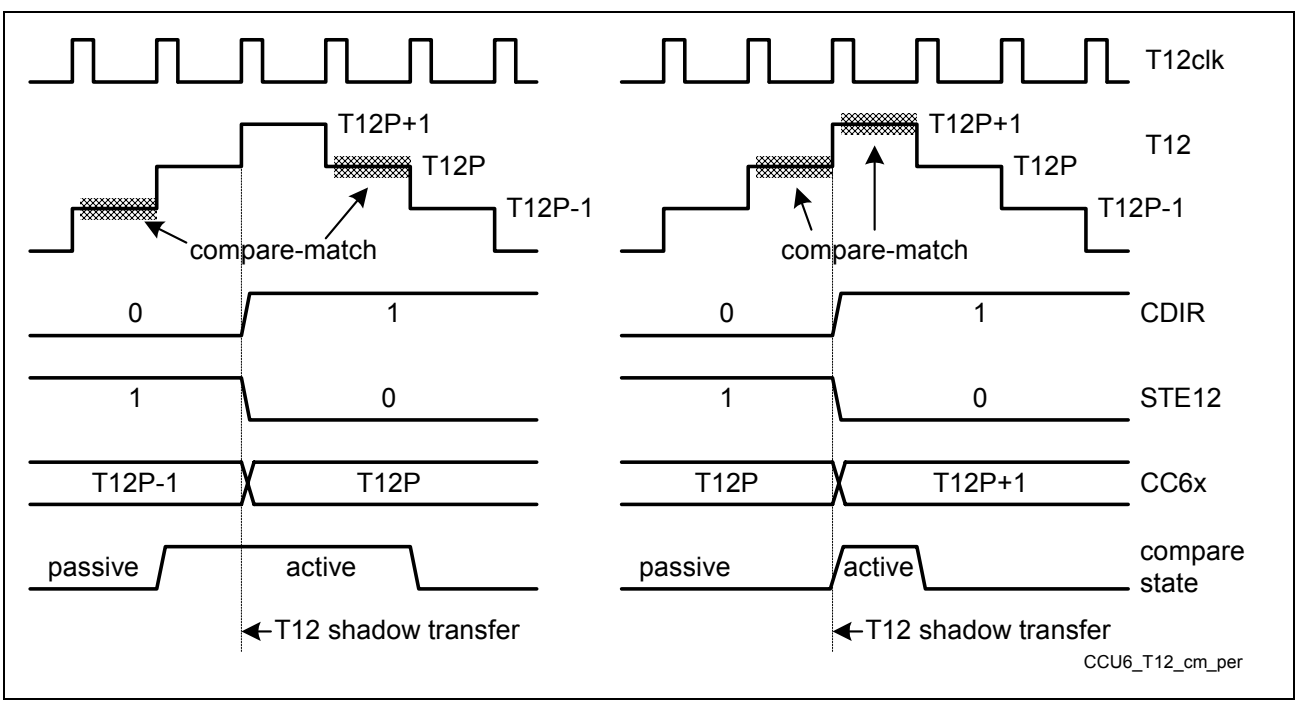


Figure 29-13 Switching Example for Duty Cycles near to 0%

29.1.2.9 Dead-time Generation

The generation of (complementary) signals for the highside and the lowside switches of one power inverter phase is based on the same compare channel. For example, if the highside switch should be active while the T12 counter value is above the compare value (compare state = 1), then the lowside switch should be active while the counter value is below (compare state = 0). The compare state, which may lead to an active output (respecting other modulation sources and the trap functionality) can be selected by the CC6xPS bits.

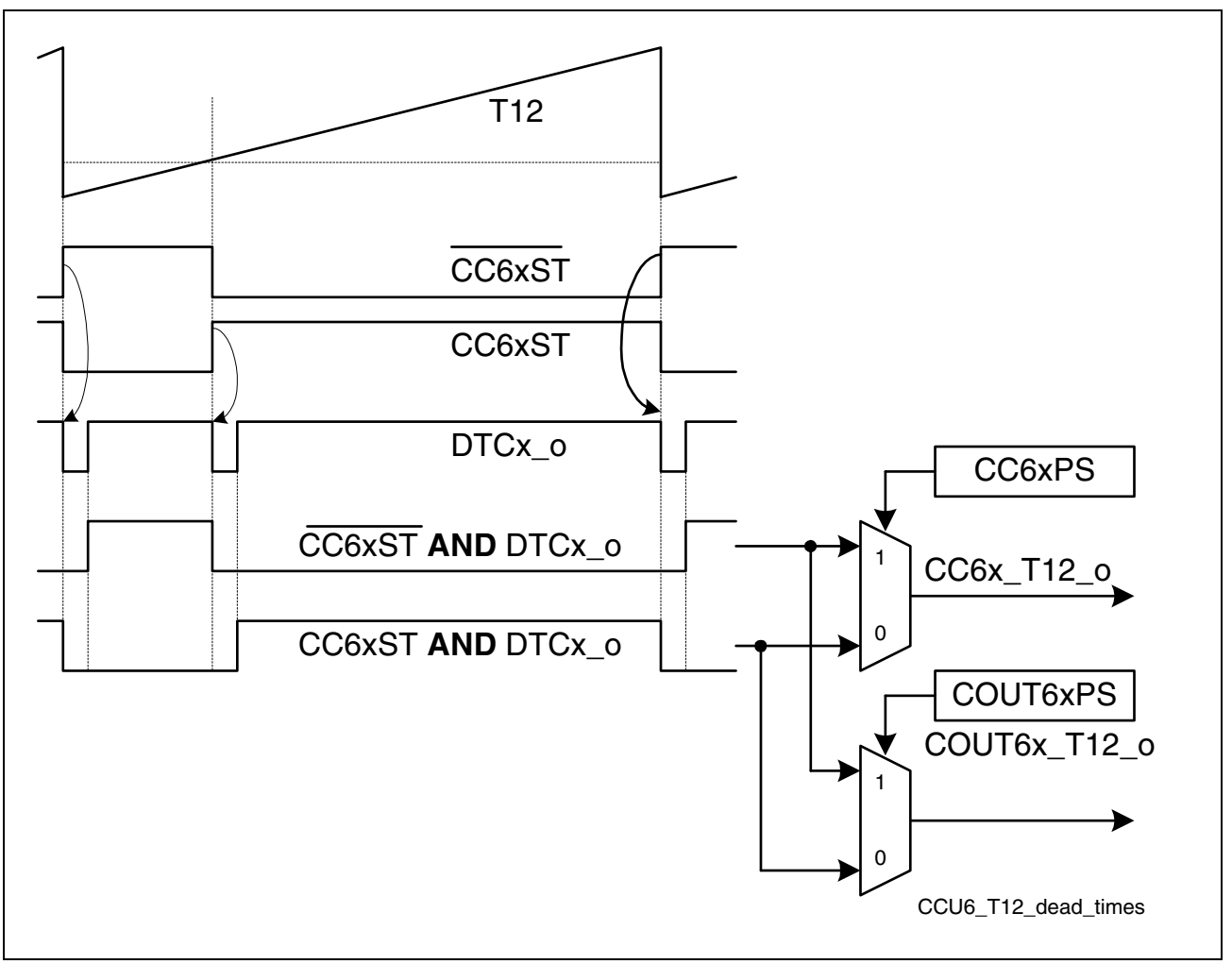


Figure 29-14 PWM-Signals with Dead-time Generation

In most cases, the switching behavior of the connected power switches is not symmetrical concerning the times needed to switch on and to switch off. A general problem arises if the time taken to switch on is less than the time to switch off the power device. In this case, a short-circuit in the inverter bridge leg occurs, which may damage the complete system. In order to solve this problem by hardware, this capture/compare unit contains a programmable dead-time counter, which delays the passive to active

edge of the switching signals (the active to passive edge is not delayed), see **Figure 29-14**.

The dead-time generation logic (see **Figure 29-15**) is built in a similar way for all three channels of T12. Each change of the CC6xST bits triggers the corresponding dead-time counter (8-bit down counter, clocked with T12clk). The trigger pulse (DTCx_rl) leads to a reload of the dead-time counter with the value, which has been programmed in bit field DTM. This reload can only take place if the dead-time feature is enabled by bit DTEx and while the counter is zero.

While counting down (zero is not yet reached), the output line DTCx_o becomes 0. This output line is combined with the T12 modulation signals, leading to a delay of the passive to active edge of the resulting signal, which is shown in **Figure 29-14**. On reaching the counter value zero, the dead-time counter stops counting and the signal DTCx_o becomes 1. The dead-time counter cannot be reloaded while it is counting.

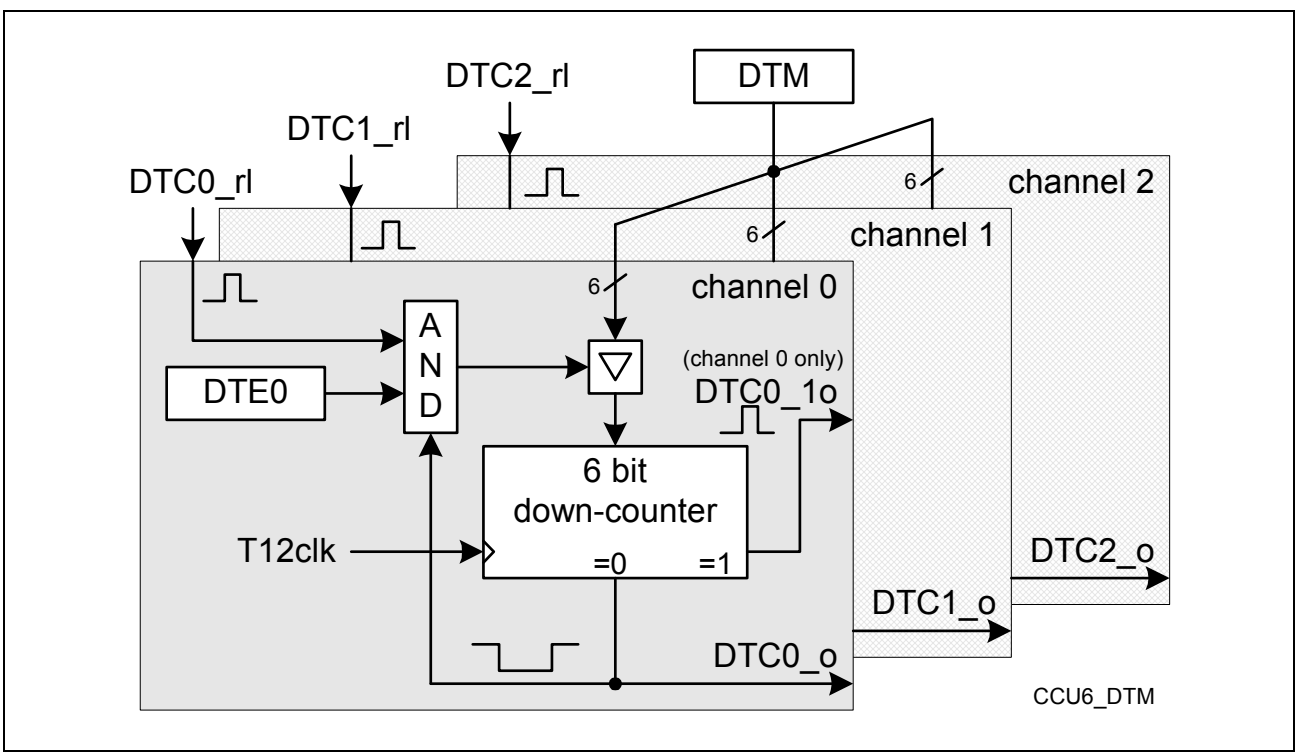


Figure 29-15 Dead-time Counter

Each of the three channels works independently with its own dead-time counter and the trigger and enable signals. The value of bit field DTM is valid for all three channels.

In the Hall Sensor mode, timer T12 is used to measure the rotational speed of the motor (channel 0 in capture mode) and to control the phase delay before switching to the next state (channel 1 in compare mode). Furthermore, channel 2 can be used to generate a time-out signal (in compare mode). As a result, T12 cannot be used for modulation and, due to the block commutation patterns, a dead-time generation is not required. In order to build an efficient noise filter for the Hall signals, channel 0 of the dead-time unit is

triggered (reloaded) with each detected edge of the Hall signals, see signal Hall_edge_o in **Figure 29-10**. For this feature, channel 0 also generates a pulse if its counter value is one.

29.1.2.10 Capture Mode

In capture mode the bits CC6xST indicate the occurrence of the selected capture event according to the bit fields MSEL6x. A rising and/or a falling edge on the pins CC6x can be selected as capture event, that is used to transfer the contents of timer T12 to the CC6xR and CC6xSR registers. In order to work in capture mode, the capture pins must be configured as inputs.

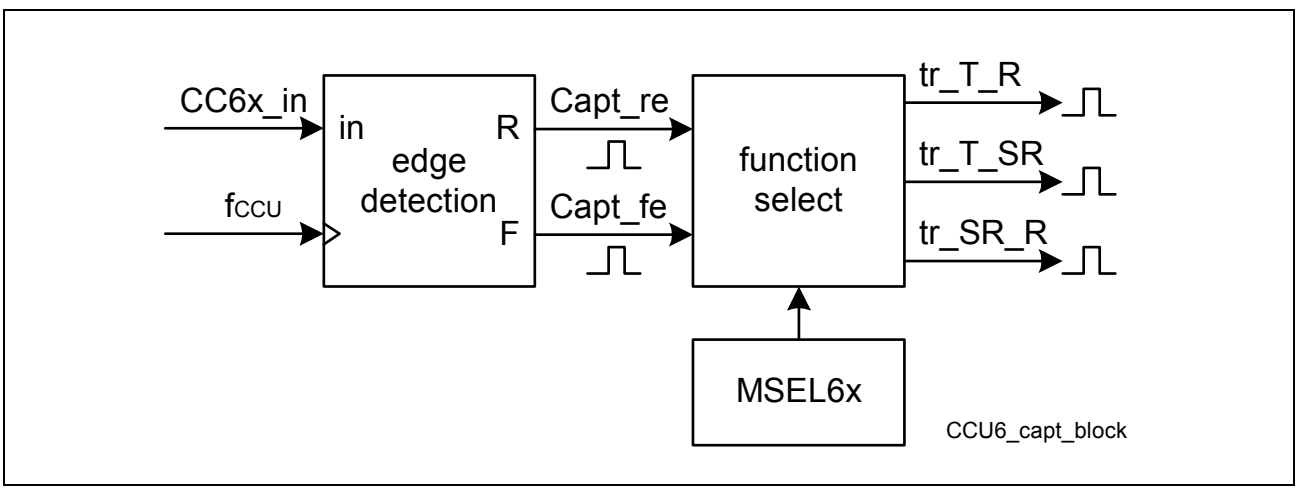


Figure 29-16 Capture Logic

The block diagram of the capture logic for one channel is shown in **Figure 29-16**. This logic is identical for all three independent channels of timer T12. The input signal (CC6x_in) from the input pin CC6x is connected to an edge detection logic delivering two output signals, one for the rising edge (Capt_re) and one for the falling edge (Capt_fe). These signals are also used as trigger sources for the channel interrupts if capture mode is selected.

There are several possibilities to store the captured values in the registers. In double register capture mode, the timer value is stored in the channel shadow register CC6xSR. The value formerly stored in this register is simultaneously copied to the channel register CC6xR. This mode can be used if two capture events occur with very little time between them. The software can then check the newly captured value while still preserving the possibility of reading the value captured earlier.

The selection of the capture mode is done by bit field MSEL6x. According to the selected mode and the detected capture event, the signals tr_T_R (transfer T12 contents to register CC6xR), tr_T_SR (transfer T12 contents to register CC6xSR) or tr_SR_R (transfer contents of CC6xSR to register CC6xR) are activated.

Note: In capture mode, a shadow transfer can be requested according to the shadow transfer rules, except for the capture/compare registers, that are left unchanged.

29.1.2.11 Single Shot Mode

In single shot mode, the timer T12 stops automatically at the end of its counting period. **Figure 29-17** shows the functionality at the end of the timer period in edge-aligned and in center-aligned modes. If the end of period event is detected while bit T12SSC is set, the bits T12R and all CC6xST bits are reset.

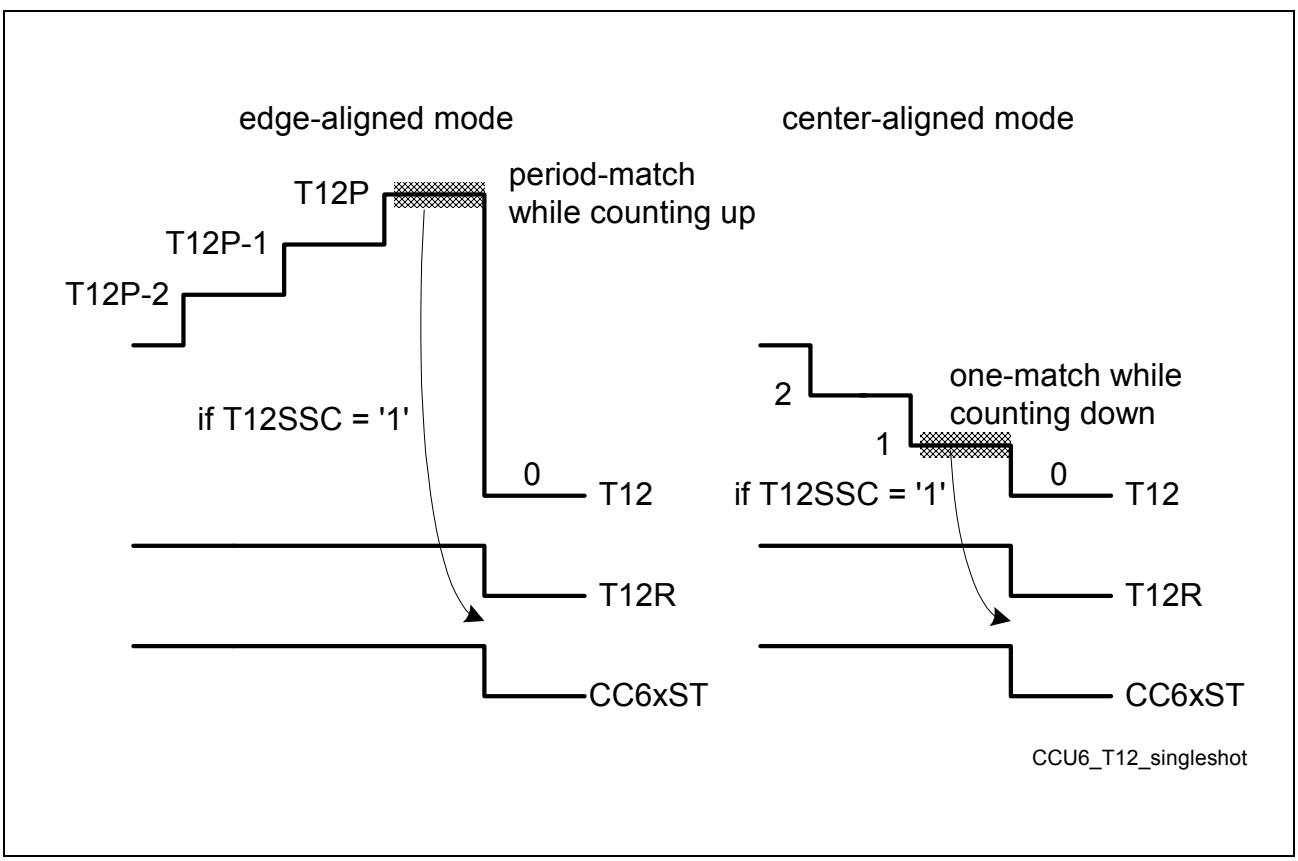


Figure 29-17 End of Single Shot Mode of T12

29.1.2.12 Hysteresis-Like Control Mode

The hysteresis-like control mode (MSEL6x = 1001_B) offers the possibility of switching off the PWM output if the input CCPOSx becomes 0 by resetting bit CC6xST. This can be used as a simple motor control feature by using a comparator to indicate, for example, overcurrent. While CCPOSx = 0, the PWM outputs of the corresponding channel are driving their passive levels. The setting of bit CC6xST is only possible while CCPOSx = 1.

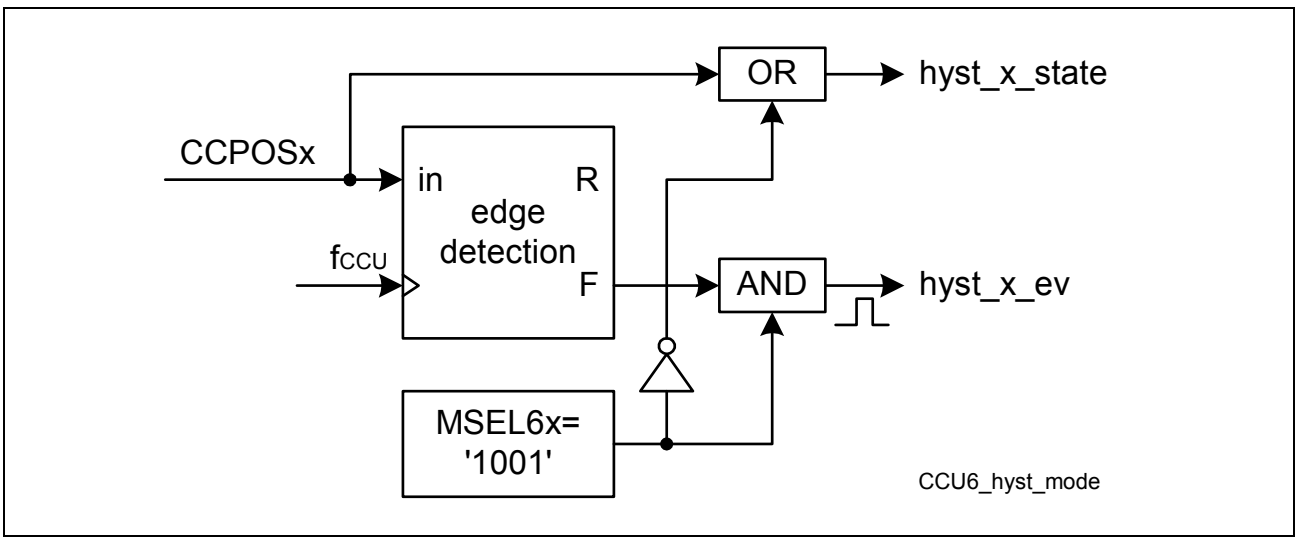


Figure 29-18 Hysteresis-Like Control Mode Logic

This mode can be used to introduce a timing-related behavior to a hysteresis controller. A standard hysteresis controller detects if a value exceeds a limit and switches its output according to the compare result. Depending on the operating conditions, the switching frequency and the duty cycle are not fixed, but change permanently.

If (outer) control loops based on a hysteresis controller (inner loop) should be implemented, the outer loops show a better behavior if they are synchronized to the inner loops. Therefore, the hysteresis-like mode can be used, which combines timer-related switching with a hysteresis controller behavior. For example, in this mode an output can be switched on a fixed time base, but it is switched off as soon as a rising edge is detected at input CCPOSx.

29.1.3 Timer T13

29.1.3.1 Overview

The timer T13 is built similar to T12, but only with one channel in compare mode. The counter can only count up (similar to the edge-aligned mode of T12). The T13 shadow transfer in case of a period-match is enabled by bit STE13 in register TCTR0. During the T13 shadow transfer, the contents of register CC63SR is transferred to register CC63R. Both registers can be read by software, whereas only the shadow register can be written by software.

The bits CC63PS, T13IM and PSL63 have shadow bits. The contents of the shadow bits are transferred to the actually used bits during the T13 shadow transfer. Write actions target the shadow bits, read actions deliver the value of the actually used bit.

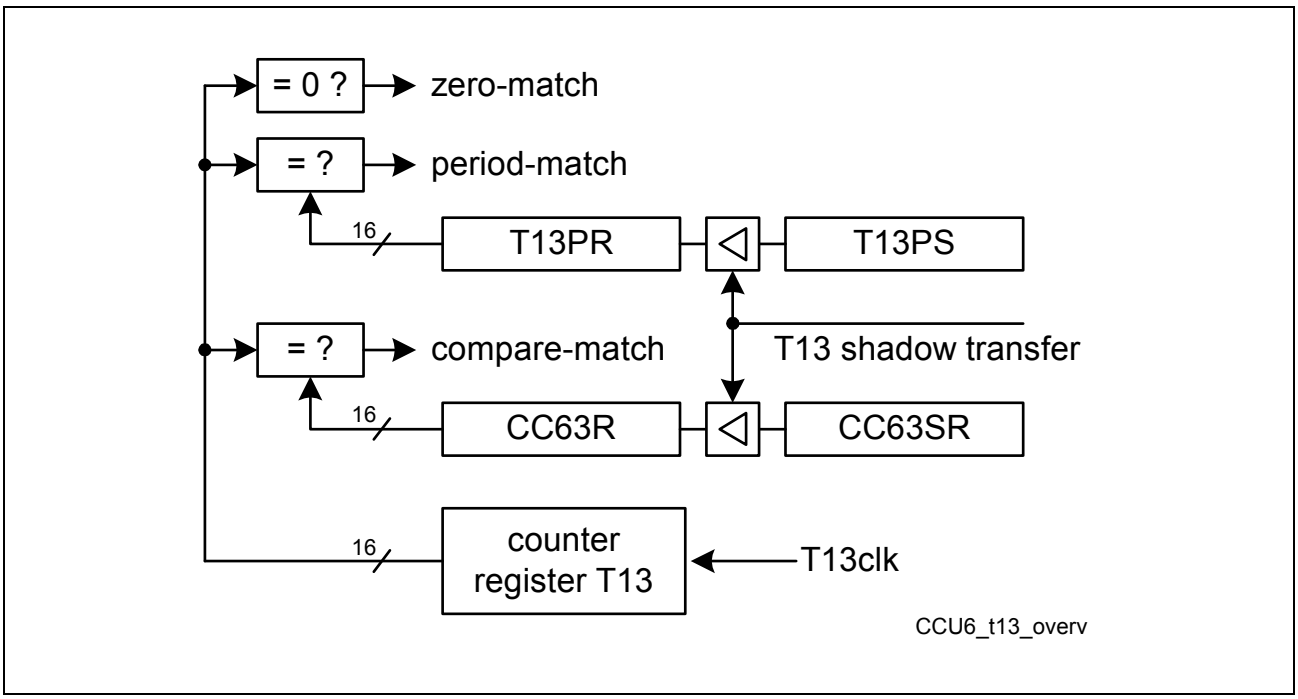


Figure 29-19 T13 Overview

Timer T13 counts according to the same counting and switching rules as timer T12 in edge-aligned mode.

29.1.3.2 Compare Mode

The compare structure of T13 is based on the compare signals T13_ST_se (compare - match detected) and T13_ST_re (zero-match detected without compare-match). These compare signals may modify bit CC63ST only while the timer is running (T13R = 1).

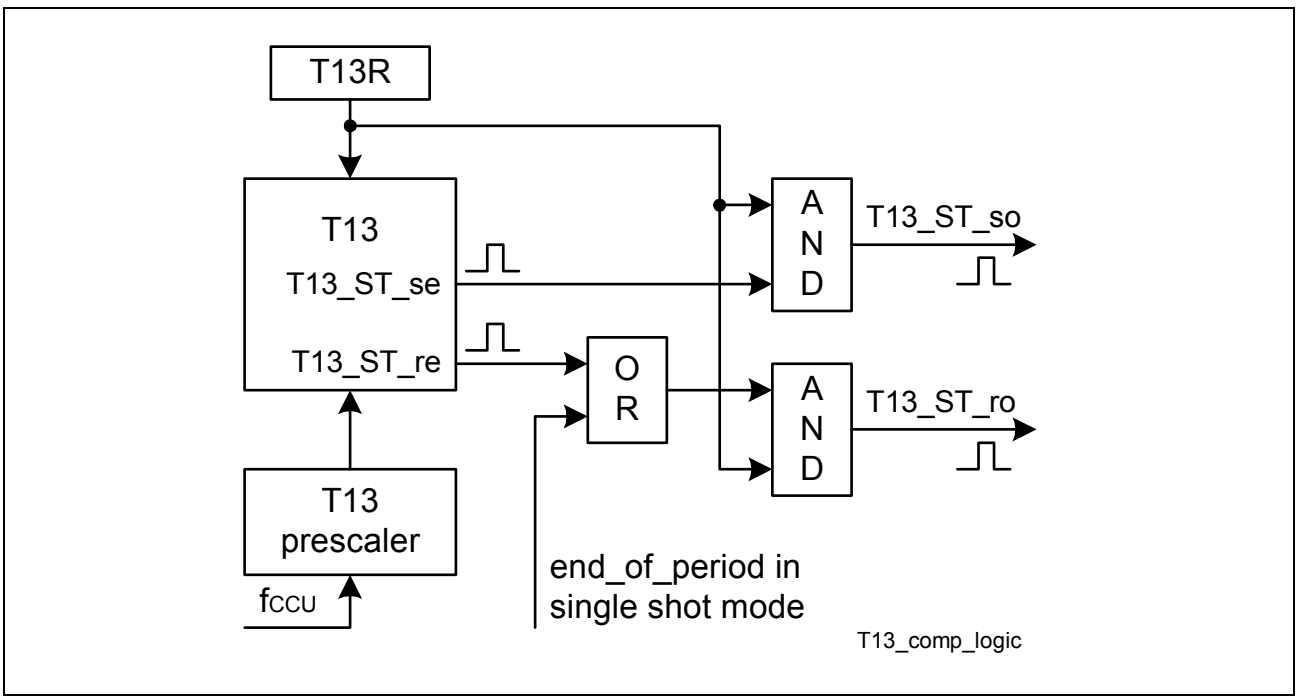


Figure 29-20 T13 Compare Logic

Similar to T12, bit CMPSTAT.CC63ST can be modified by software by CC63SR and CC63R register. The output line COUT63_T13_o can generate a T13 PWM at the output pin COUT63. The signal MOD_T13_o can be used to modulate the other output signals with a T13 PWM. In order to decouple COUT63 from the internal modulation, the compare state leading to an active signal can be selected independently by bits CMPSTAT.T13IM and CMPSTAT.COUT63PS.

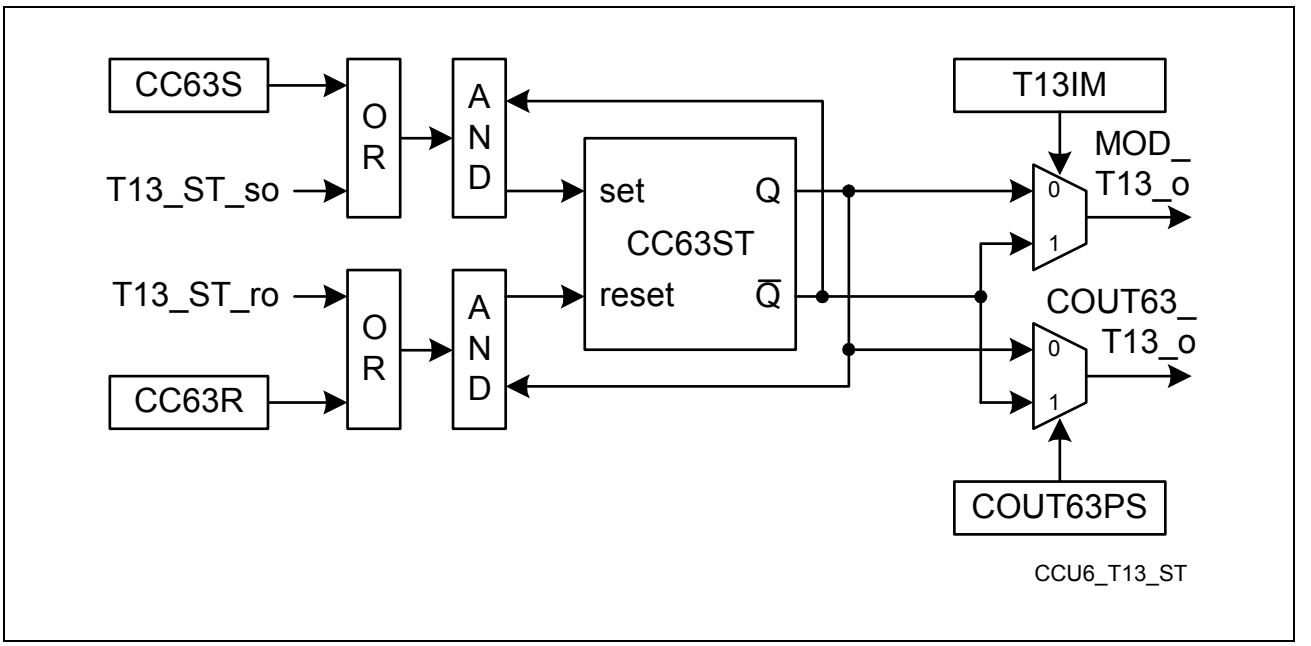


Figure 29-21 T13 Logic for CC6xST Control

29.1.3.3 Single Shot Mode

The single shot mode of T13 is similar to the single shot mode of T12 in edge-aligned mode.

29.1.3.4 External Timer Start

The external timer start feature of T13 is similar to the one of T12.

29.1.3.5 Synchronization of T13 to T12

The timer T13 can be synchronized on a T12 event. The bit fields TCTR2.T13TEC and TCTR2.T13TED select the event, which is used to start timer T13. This event sets bit T13R per hardware and T13 starts counting. Combined with the single shot mode, this feature can be used to generate a programmable delay after a T12 event.

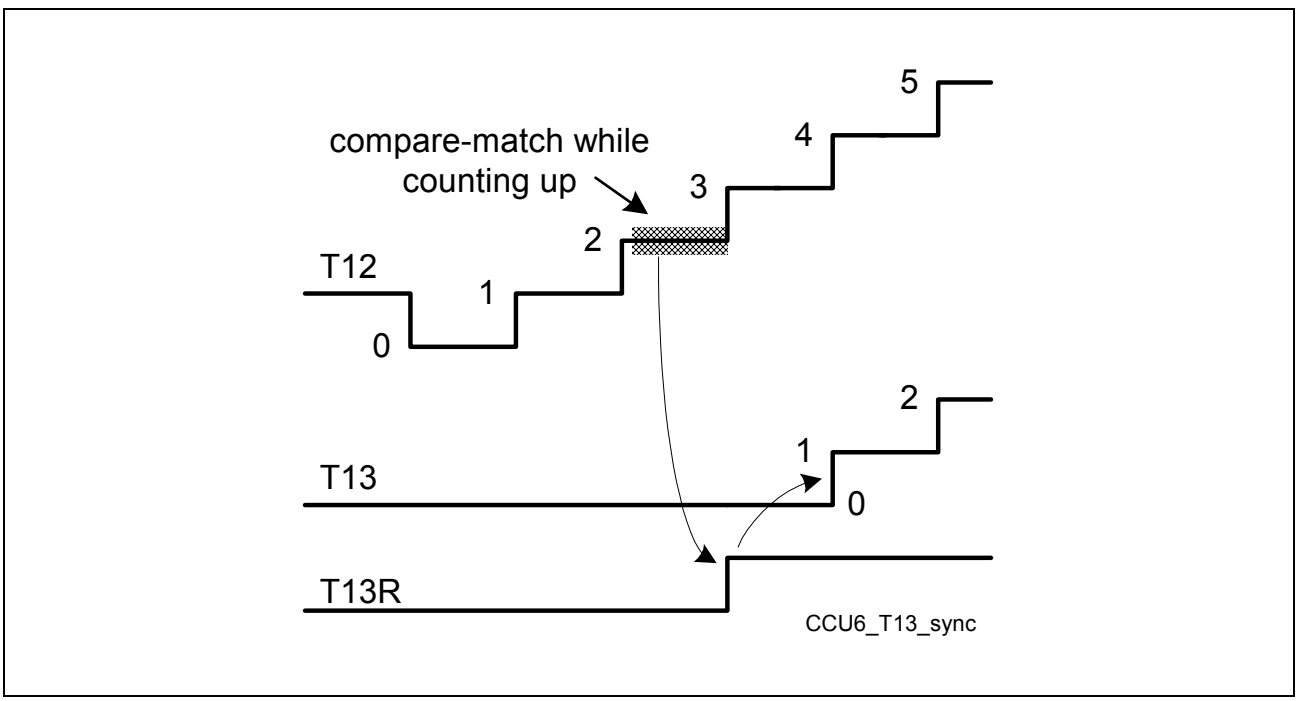


Figure 29-22 Synchronization of T13 to T12

Figure 29-22 shows the synchronization of T13 to a T12 event. The selected event in this example is a compare-match (compare value = 2) while counting up. The clocks of T12 and T13 can be different (other prescaler factor), but for reasons of simplicity, this example shows the case for T12clk equal to T13clk.

29.1.4 Modulation Control

The modulation control part combines the different modulation sources (CC6x_T12_o, COUT6x_T12_o = six T12-related signals from the three compare channels), the T13-related signal (MOD_T13_o) and the multi-channel modulation signals (MCMP bits). Each modulation source can be individually enabled for each output line. Furthermore, the trap functionality is taken into account to disable the modulation of the corresponding output line during the trap state (if enabled).

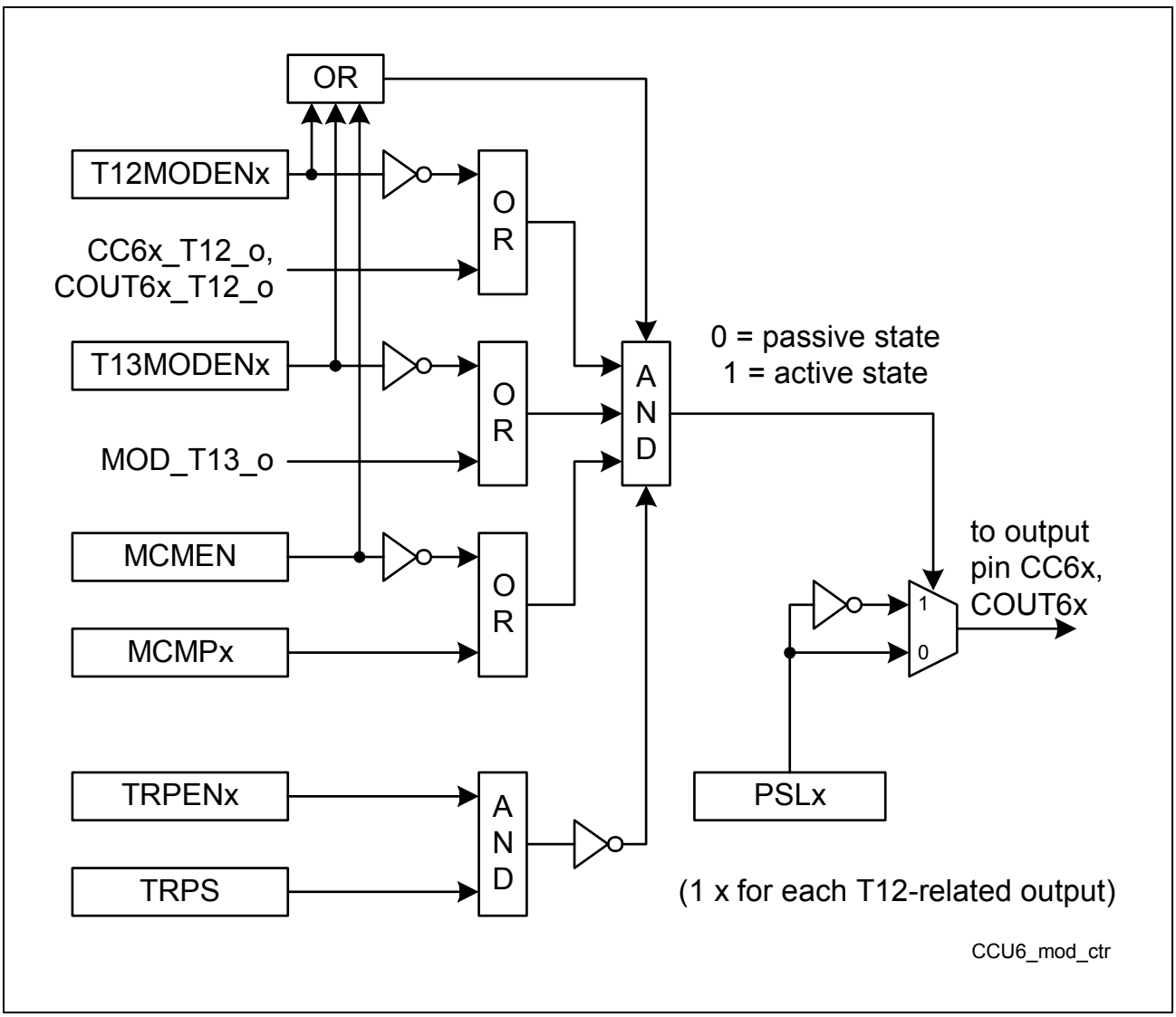


Figure 29-23 Modulation Control of T12-related Outputs

The logic shown in **Figure 29-23** must be built separately for each of the six T12-related output lines, referring to the index ‘x’ in the figure above.

The output level that is driven while the output is in the passive state is defined by the corresponding bit PSLR.PSL. If the resulting modulation signal is active, the inverted level of the PSLx bit is driven by the output stage.

The modulation control part for the T13-related output COUT63 combines the T13 output signal (COUT63_T13_o) and the enable bit MODCTR.ECT13O with the trap functionality. The output level of the passive state is selected by bit PSLR.PSL63.

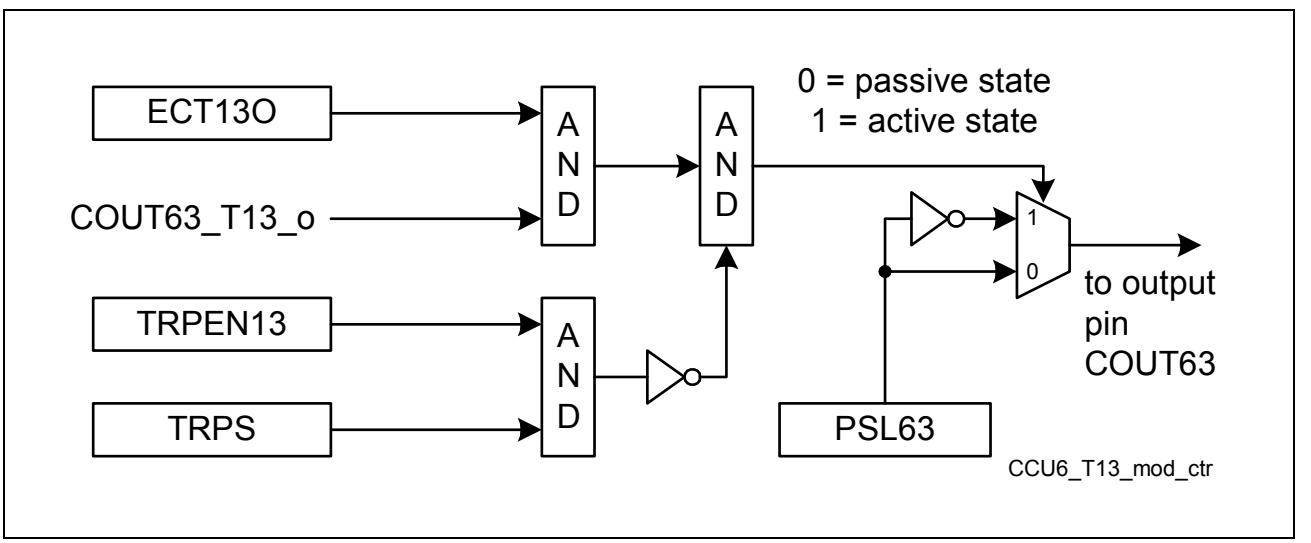


Figure 29-24 Modulation Control of the T13-related Output COUT63

Note: In order to avoid spikes on the output lines, the seven output signals (CC60, COUT60, CC61, COUT61, CC62, COUT62, COUT63) are registered out with the peripheral clock.

29.1.5 Trap Handling

The trap functionality permits the PWM outputs to react on the state of the input pin $\overline{\text{CTRAP}}$. This functionality can be used to switch off the power devices if the trap input becomes active (e.g. as emergency stop).

During the trap state, the selected outputs are forced to the passive state and no active modulation is possible. The trap state is entered immediately by hardware if the $\overline{\text{CTRAP}}$ input signal becomes active and the trap function is enabled by bit TRPCTR.TRPPEN. It can also be entered by software by setting bit IS.TRPF (trap input flag), leading to IS.TRPS = 1 (trap state indication flag). The trap state can be left when the input is inactive, by software control and synchronized to the following events:

- TRPF is automatically reset after $\overline{\text{CTRAP}}$ becomes inactive
  (if TRPCTR.TRPM2 = 0)
- TRPF must be reset by software after $\overline{\text{CTRAP}}$ becomes inactive
  (if TRPCTR.TRPM2 = 1)
- Synchronized to T12 PWM after TRPF is reset
  (T12 period-match in edge-aligned mode or one-match while counting down in center-aligned mode)
- Synchronized to T13 PWM after TRPF is reset
  (T13 period-match)
- No synchronization to T12 or T13

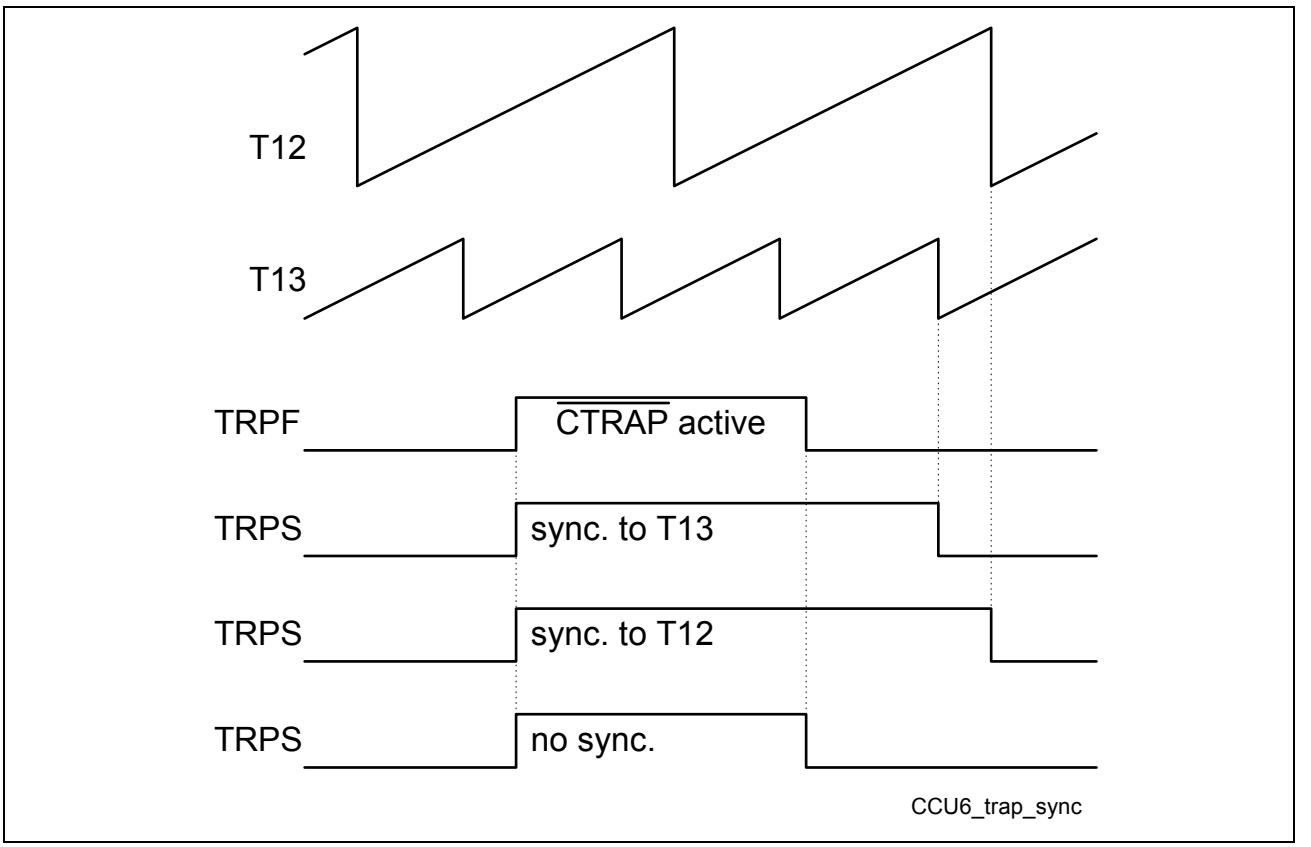


Figure 29-25 Trap State Synchronization (with TRM2 = 0)

29.1.6 Multi-Channel Mode

The multi-channel mode offers a possibility to modulate all six T12-related output signals within one instruction. The bits in bit field MCMP are used to select the outputs that may become active. If the multi-channel mode is enabled (bit MCMEN = 1), only those outputs may become active, which have a 1 at the corresponding bit position in bit field MCMP.

This bit field has its own shadow bit field MCMPS, which can be written by software. The transfer of the new value in MCMPS to the bit field MCMP can be triggered by and synchronized to T12 or T13 events. This structure permits the software to write the new value, which is then taken into account by the hardware at a well-defined moment and synchronized to a PWM period. This avoids unintended pulses due to unsynchronized modulation sources (T12, T13, software).

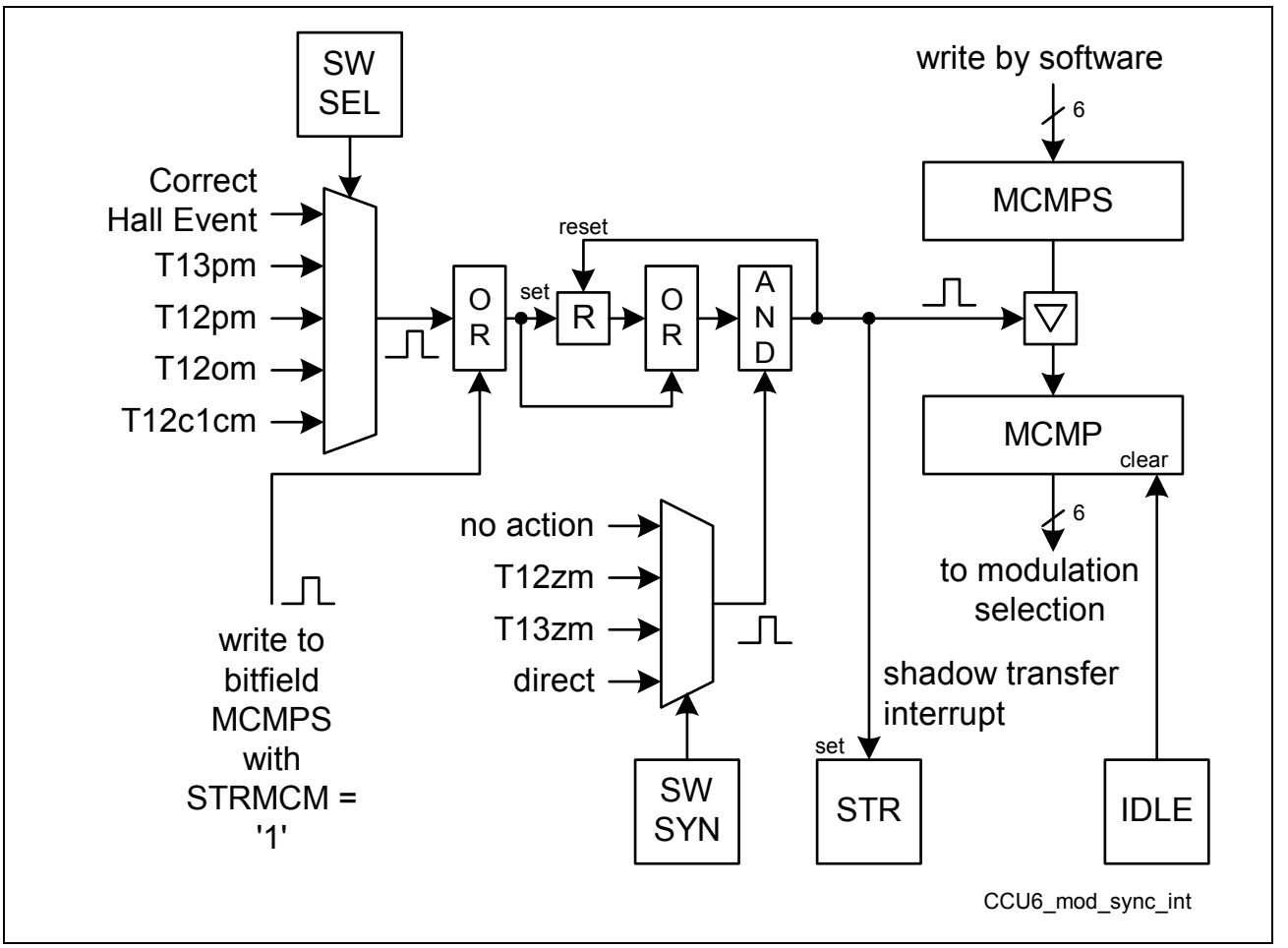


Figure 29-26 Modulation Selection and Synchronization

Figure 29-26 shows the modulation selection for the multi-channel mode. The event that triggers the update of bit field MCMP is chosen by SWSEL. If the selected switching event occurs, the reminder flag R is set. This flag monitors the update request and it is automatically reset when the update takes place. In order to synchronize the update of MCMP to a PWM generated by T12 or T13, bit field SWSYN allows the selection of the synchronization event, which leads to the transfer from MCMPS to MCMP. Due to this structure, an update takes place with a new PWM period.

If it is explicitly desired, the update takes place immediately with the setting of flag R when the direct synchronization mode is selected. The update can also be requested by software by writing to bit field MCMPS with the shadow transfer request bit STRMCM set. If this bit is set during the write action to the register, the flag R is automatically set. By using the direct mode and bit STRMCM, the update takes place completely under software control.

A shadow transfer interrupt can be generated when the shadow transfer takes place.

The possible hardware request events are:

- a T12 period-match while counting up (T12pm)

- a T12 one-match while counting down (T12om)
- a T13 period-match (T13pm)
- a T12 compare-match of channel 1 (T12c1cm)
- a correct Hall event

The possible hardware synchronization events are:

- a T12 zero-match while counting up (T12zm)
- a T13 zero-match (T13zm)

29.1.7 Hall Sensor Mode

29.1.7.1 Introduction

In **Brushless-DC motors** the next multi-channel state values depend on the pattern of the Hall inputs. There is a strong correlation between the **Hall pattern** (CURH) and the **modulation pattern** (MCMP). Because of different machine types, the modulation pattern for driving the motor can be different. Therefore, it is advantageous to have a wide flexibility in defining the correlation between the Hall pattern and the corresponding modulation pattern. The CCU6 offers this by having a register which contains the actual Hall pattern (CURHS), the next expected Hall pattern (EXPHS) and its output pattern (MCMPS). At every correct Hall event (CHE, see **Figure 29-28**) a new Hall pattern with its corresponding output pattern can be loaded (from a predefined table) by software into the register MCMOUTS. Loading this shadow register can also be done by a write action on MCMOUTS with bit STRHP = 1. In case of a phase delay (generated by T12 channel 1) a new pattern can be loaded when the multi-channel mode shadow transfer (indicated by bit STR) occurs.

29.1.7.2 Sampling of the Hall Pattern

The **sampling** of the Hall pattern (on CCPOSx) is with the module clock f_{CCU}. By using the dead-time counter DTC0 (mode MSEL6x = 1000_B) a hardware **noise filter** can be implemented to suppress spikes on the Hall inputs due to high di/dt in rugged inverter environment. In case of a Hall event the DTC0 is reloaded, starts counting and generates a delay between the detected event and the sampling point. After the counter value of one is reached, the CCPOSx inputs are sampled (without noise and spikes) and are compared to the current Hall pattern (CURH) and to the expected Hall pattern (EXPH). If the sampled pattern equals to the current pattern, it means that the edge on CCPOSx was due to a noise spike and no action will be triggered (implicit noise filter by delay). If the sampled pattern equals to the next expected pattern, the edge on CCPOSx was a correct Hall event, the bit CHE is set which causes an interrupt.

Additionally, if enabled by all MSEL6x bit fields, timer T12 can be controlled for Hall mode specific actions when a correct Hall event is detected. These specific actions are the capturing of the current contents of T12 in register CC60R and a reset of T12 (for new a speed measurement). Furthermore, a shadow transfer takes place for the compare channels CC61 and CC62 (for phase delay generation and time-out criteria).

In the case that the multi-channel mode and the Hall pattern comparison should work independently from timer T12, the delay generation by DTC0 can be bypassed. In this case, timer T12 can be used for other purposes. In order to increase flexibility, the signal to start a Hall pattern comparison (hcrdy) can be selected among several sources, see **Figure 29-27**.

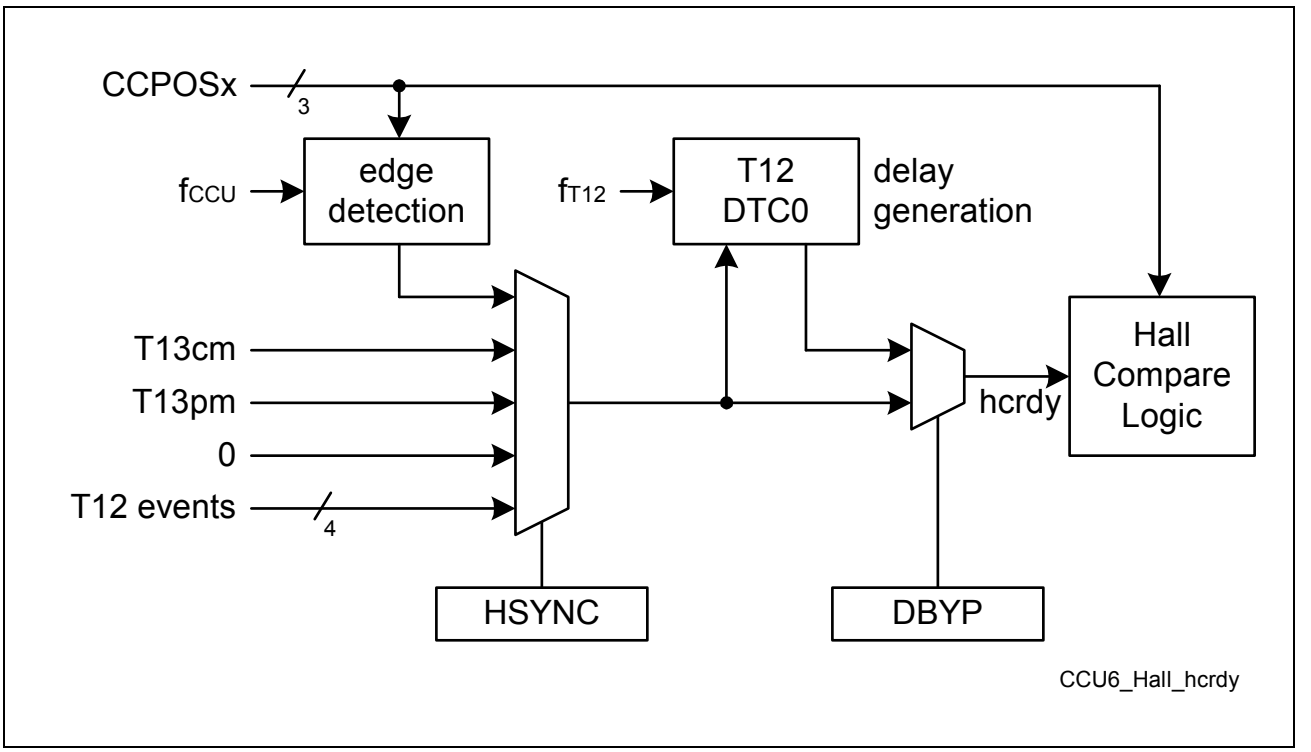


Figure 29-27 Trigger for Hall Compare

29.1.7.3 Hall Events

This correct Hall event can be used as a transfer request event for register MCMOUTS. The transfer from MCMOUTS to MCMOUT transfers the new CURH-pattern as well as the next EXPH-pattern. In case the sampled Hall inputs were neither the current nor the expected Hall pattern, the bit WHE (wrong Hall event) is set which can also cause an interrupt and set the IDLE mode clearing MCMP (modulation outputs are inactive). To restart from IDLE, the transfer request of MCMOUTS must be initiated by software (bit STRHP and bit fields SWSEL/SWSYN).

29.1.7.4 Hall Compare Logic

The logic for the Hall compare action (correct Hall event, wrong Hall event) is shown in **Figure 29-28**. It shows the functional dependencies when the Hall compare action is triggered (by signal hcrdy = Hall compare ready).

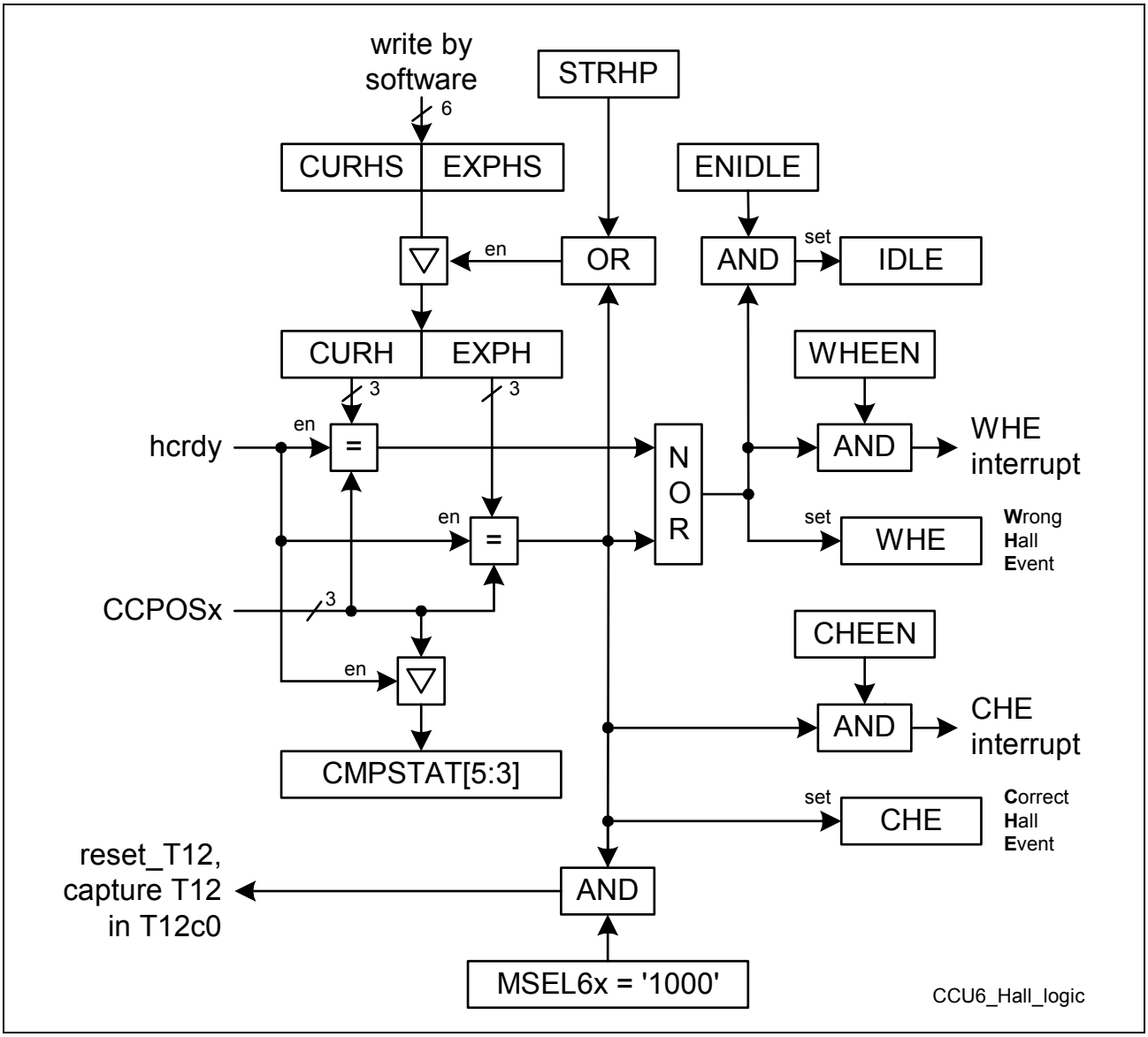


Figure 29-28 Hall Compare Logic

29.1.7.5 Brushless-DC Control

For **Brushless-DC** motors there is a special mode (MSEL6x = 1000_B) which is triggered by a change of the Hall-inputs (CCPOSx). This mode shows the capabilities of the CCU6 (see **Figure 29-23**, **Figure 29-26**, **Figure 29-28** and **Figure 29-29**). Here, T12's channel 0 acts in capture function, channel 1 and 2 in compare function (without output modulation) and the multi-channel-block is used to trigger the output switching together with a possible modulation of T13.

After the detection of a valid Hall edge the T12 count value is captured to channel 0 (representing the actual motor speed) and resets the T12. When the timer reaches the compare value in channel 1, the next multi-channel state is switched by triggering the shadow transfer of bit field MCMP (if enabled in bit field SWEN). This trigger event can be combined with several conditions which are necessary to implement a noise filtering (correct Hall event) and to synchronize the next multi-channel state to the modulation sources (avoiding spikes on the output lines). This compare function of channel 1 can be used as a phase delay for the position input to the output switching which is necessary if a sensorless back-EMF technique is used instead of Hall sensors. The compare value in channel 2 can be used as a time-out trigger (interrupt) indicating that the motors destination speed is far below the desired value which can be caused by an abnormal load change. In this mode, the modulation of T12 must be disabled (T12MODENx = 0).

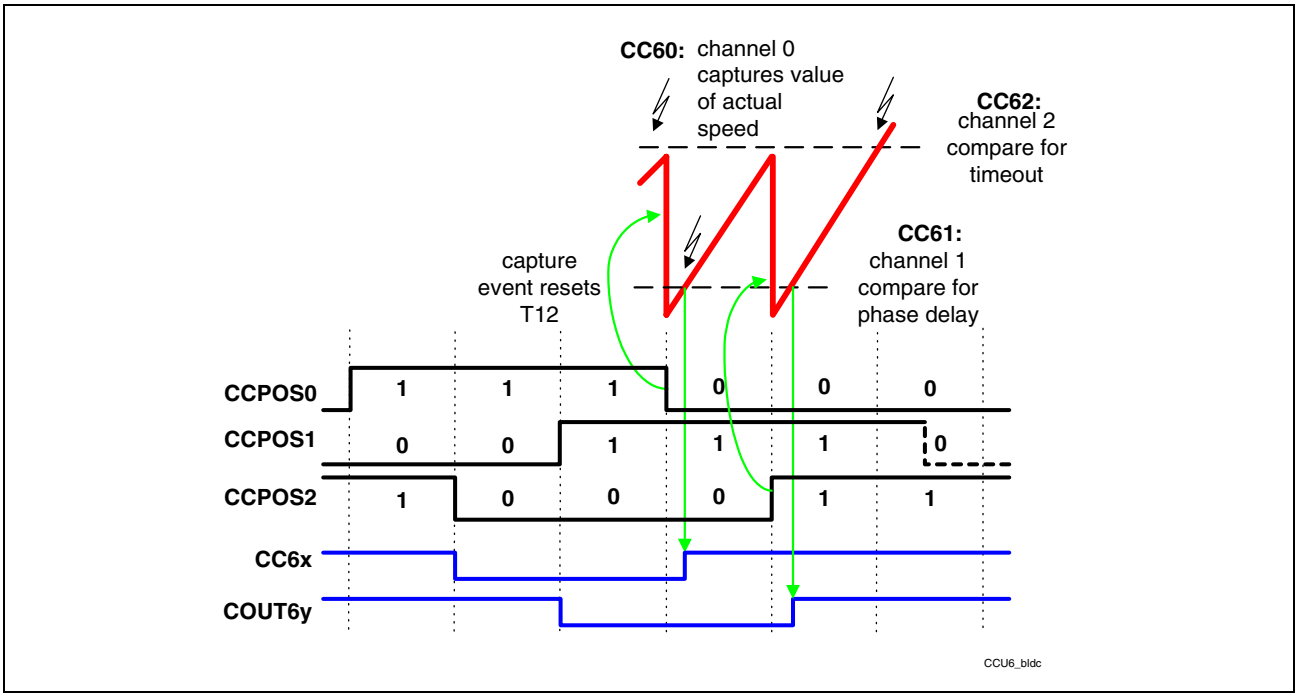


Figure 29-29 Timer T12 Brushless-DC Mode (all MSEL6x = 1000_H)

The capturing of the timer value in register CC60R, the shadow transfers from registers CC61SR to CC61R, from CC62SR to CC62R, and shadow transfer for the T12 period value are done together with the reset event for T12 (according to the value of the STE bit).

29.1.8 Interrupt Generation

The interrupt structure is shown in Figure 29-30. The interrupt event or the corresponding interrupt set bit (in register ISS) can trigger the interrupt generation. The interrupt pulse is generated independently from the interrupt flag in register IS. The interrupt flag can be reset by software by writing to the corresponding bit in register ISR.

If enabled by the related interrupt enable bit in register IEN, an interrupt pulse can be generated at one of the four interrupt output lines of the module.

If more than one interrupt source is connected to the same interrupt node pointer (in register INP), the requests are combined to one common line.

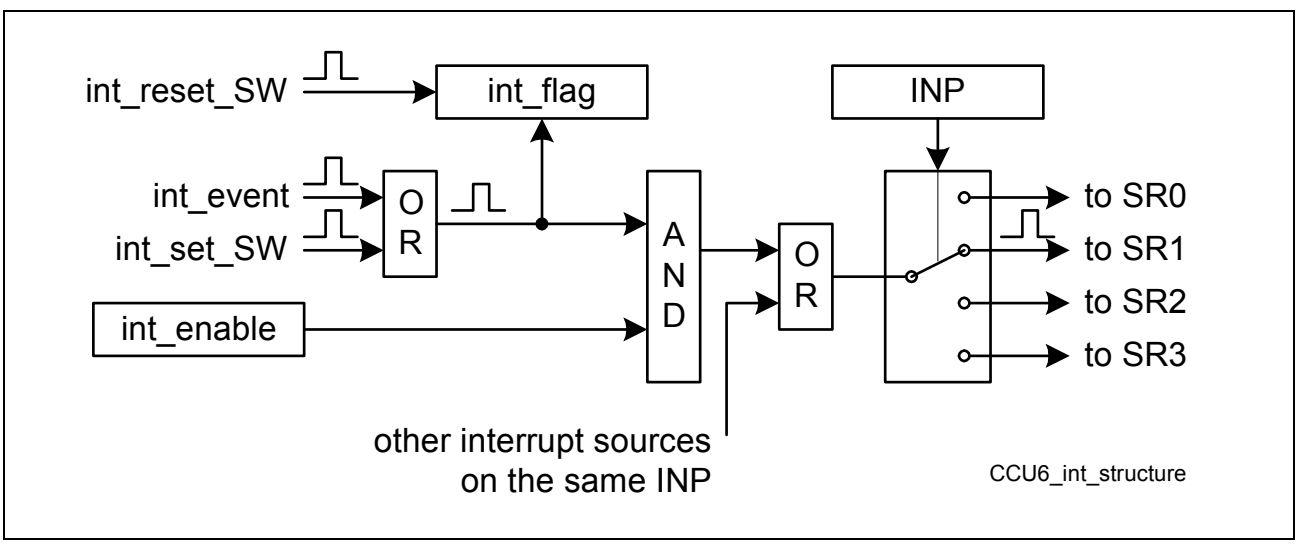


Figure 29-30 Interrupt Generation

The interrupt sources of the CCU6 module can be mapped to four interrupt output lines by programming the interrupt node pointer register INP. The default assignment of the interrupt sources to the output lines and their corresponding control registers is listed in Table 29-1:

Table 29-1 CCU6 Default Interrupt Node Assignment

Source of Interrupt	Interrupt Output Line	Service Request Control Register
Channel 0 Interrupts	SRC0	CCU6_SRC0
Channel 1 Interrupts	SRC0	CCU6_SRC0
Channel 2 Interrupts	SRC0	CCU6_SRC0
Correct Hall Pattern Interrupts	SRC1	CCU6_SRC1
Emergency Interrupts	SRC1	CCU6_SRC1

Table 29-1 CCU6 Default Interrupt Node Assignment (cont'd)

Source of Interrupt	Interrupt Output Line	Service Request Control Register
Timer T12 Interrupts	SRC2	CCU6_SRC2
Timer T13 Interrupts	SRC3	CCU6_SRC3

29.1.9 Suspend Mode

In suspend mode, the functional clock f_{CCU} of the module kernel is stopped. The registers can still be accessed by the CPU (read and write). This mode is useful for debugging purposes, e.g. where the current device status should be 'frozen' in order to get a snapshot of the internal values.

The suspend mode can be entered when the suspend mode is requested, the suspend mode is enabled and the module has reached a safe, deterministic state (equal to the timer stop conditions in single shot mode). This behavior avoids critical situations if a power inverter is connected to the module's outputs.

The suspend mode is non-intrusive concerning the register bits. Register bits must not be changed by hardware when entering or leaving the suspend mode. In suspend mode, all registers can be accessed by write or read instructions for debugging purposes.

In suspend mode, the timers T12 and T13 are not running. Other module functions are still available. The suspend request can lead to a behavior of the output signals equivalent to the trap case.

29.2 CCU6 Kernel Registers

Figure 29-31 and Table 29-2 show all registers associated with the CCU6 Kernel.

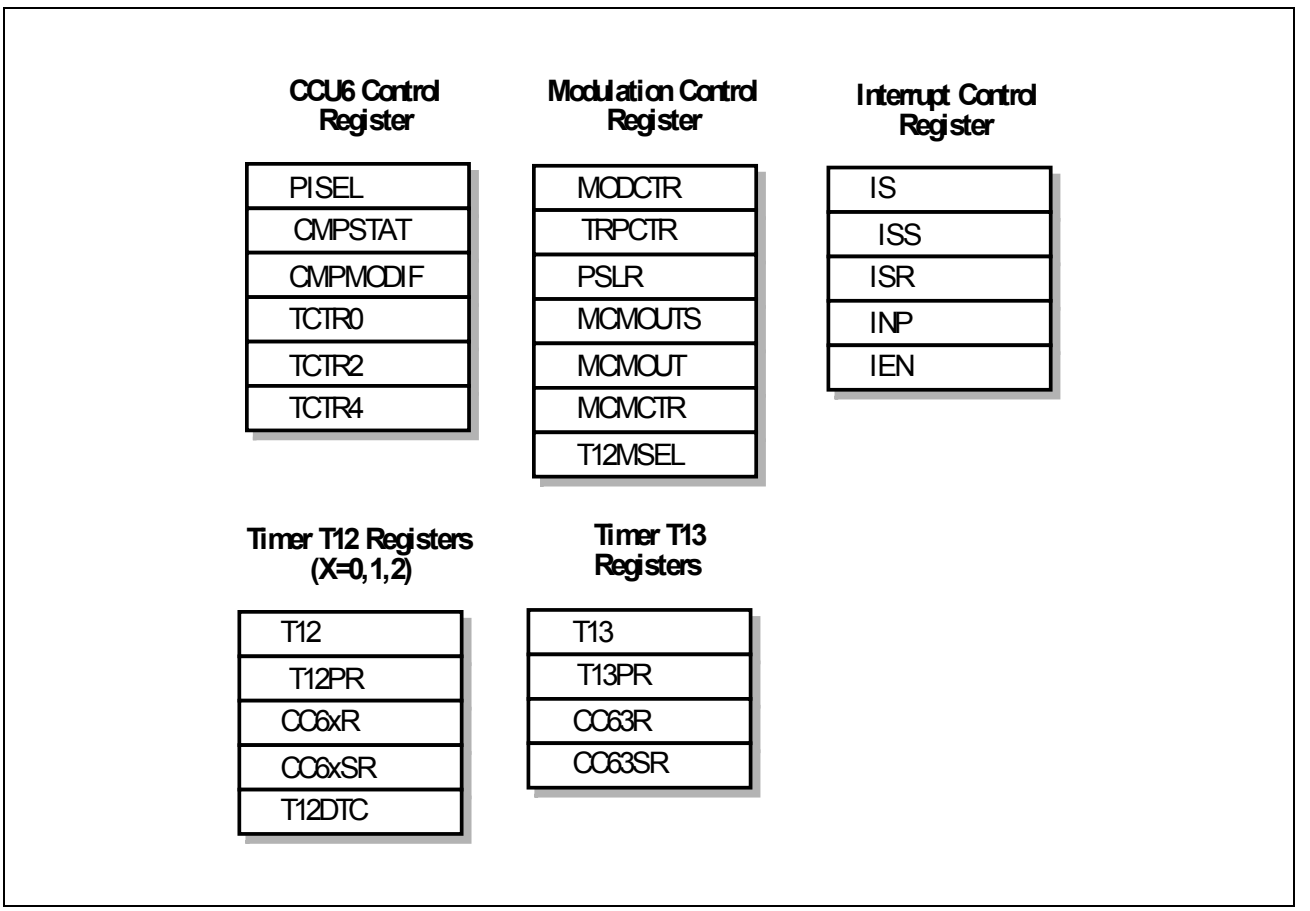


Figure 29-31 CCU6 Register Overview

Note: If a hardware and a software request to modify a bit occur simultaneously, the software wins.

Table 29-2 CCU Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
PISEL0	Port Input Select Register 0	0010_H	Page 29-38
PISEL2	Port Input Select Register 2	0014_H	Page 29-38
T12	Timer T12 Counter Register	0020_H	Page 29-52
T12PR	Timer T12 Period Register	0024_H	Page 29-53
T12DTC	Timer T12 Dead-Time Control Register	0028_H	Page 29-56
CC60R	Capture/Compare Register T12 Channel 0	0030_H	Page 29-54
CC61R	Capture/Compare Register T12 Channel 1	0034_H	Page 29-54

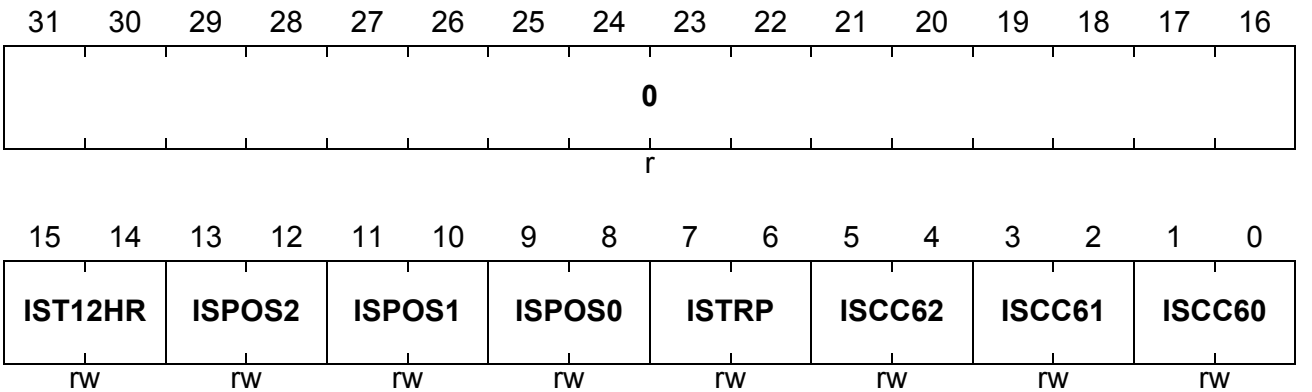
Table 29-2 CCU Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
CC62R	Capture/Compare Register T12 Channel 2	0038_H	Page 29-54
CC60SR	Compare Shadow Register T12 Channel 0	0040_H	Page 29-55
CC61SR	Compare Shadow Register T12 Channel 1	0044_H	Page 29-55
CC62SR	Compare Shadow Register T12 Channel 2	0048_H	Page 29-55
T13	Timer T13 Counter Register	0050_H	Page 29-58
T13PR	Timer T13 Period Register	0054_H	Page 29-59
CC63R	Compare Register T13	0058_H	Page 29-60
CC63SR	Compare Shadow Register T13	$005C_H$	Page 29-61
CMPSTAT	Compare State Register	0060_H	Page 29-41
CMPMODIF	Compare State Modification Register	0064_H	Page 29-43
TCTR0	Timer Control Register 0	0068_H	Page 29-44
TCTR2	Timer Control Register 2	$006C_H$	Page 29-47
TCTR4	Timer Control Register 4	$003C_H$	Page 29-50
MODCTR	Modulation Control Register	0070_H	Page 29-62
TRPCTR	Trap Control Register	0074_H	Page 29-64
PSLR	Passive State Level Register	0078_H	Page 29-66
T12MSEL	T12 Mode Select Register	$007C_H$	Page 29-74
MCMOUTS	Multi_Channel Mode Output Shadow Register	0080_H	Page 29-68
MCMOUT	Multi_Channel Mode Output Register	0084_H	Page 29-69
MCMCTR	Multi_Channel Mode Control Register	0088_H	Page 29-72
IS	Interrupt Status Register	0090_H	Page 29-78
ISS	Interrupt Status Set Register	0094_H	Page 29-81
ISR	Interrupt Status Reset Register	0098_H	Page 29-83
IEN	Interrupt Enable Register	$009C_H$	Page 29-85
INP	Interrupt Node Pointer Register	$00A0_H$	Page 29-89

29.2.1 CCU Control Registers

Registers PISEL0 and PISEL2 contain bit fields that select the actual input signal for the module inputs. This permits the adaptation of the pin functionality of the device to the application's requirements. The output pins are chosen according to the registers in the ports.

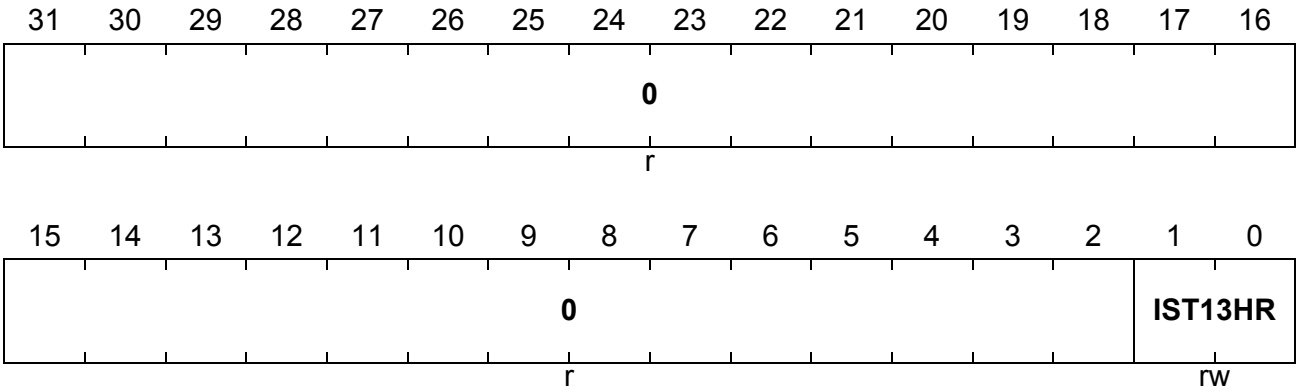
PISEL0
Port Input Select Register 0 **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
ISCC60	[1:0]	rw	**Input Select for CC60** This bit field defines the port pin, that is used for the CC60 capture input signal. 00 The input pin for CC60_I0 01 The input pin for CC60_I1 10 The input pin for CC60_I2 11 The input pin for CC60_I3
ISCC61	[3:2]	rw	**Input Select for CC61** This bit field defines the port pin, that is used for the CC61 capture input signal. 00 The input pin for CC61_I0 01 The input pin for CC61_I1 10 The input pin for CC61_I2 11 The input pin for CC61_I3
ISCC62	[5:4]	rw	**Input Select for CC62** This bit field defines the port pin, that is used for the CC62 capture input signal. 00 The input pin for CC62_I0 01 The input pin for CC62_I1 10 The input pin for CC62_I2 11 The input pin for CC62_I3

Field	Bits	Type	Description
ISTRP	[7:6]	rw	**Input Select for $\overline{\text{CTRAP}}$** This bit field defines the port pin, that is used for the $\overline{\text{CTRAP}}$ input signal. 00 The input pin for $\overline{\text{CTRAP}}$_I0 01 The input pin for $\overline{\text{CTRAP}}$_I1 10 The input pin for $\overline{\text{CTRAP}}$_I2 11 The input pin for $\overline{\text{CTRAP}}$_I3
ISPOS0	[9:8]	rw	**Input Select for CCPOS0** This bit field defines the port pin, that is used for the CCPOS0 input signal. 00 The input pin for CCPOS0_I0 01 The input pin for CCPOS0_I1 10 The input pin for CCPOS0_I2 11 The input pin for CCPOS0_I3
ISPOS1	[11:10]	rw	**Input Select for CCPOS1** This bit field defines the port pin, that is used for the CCPOS1 input signal. 00 The input pin for CCPOS1_I0 01 The input pin for CCPOS1_I1 10 The input pin for CCPOS1_I2 11 The input pin for CCPOS1_I3
ISPOS2	[13:12]	rw	**Input Select for CCPOS2** This bit field defines the port pin, that is used for the CCPOS2 input signal. 00 The input pin for CCPOS2_I0 01 The input pin for CCPOS2_I1 10 The input pin for CCPOS2_I2 11 The input pin for CCPOS2_I3
IST12HR	[15:14]	rw	**Input Select for T12HR** This bit field defines the port pin, that is used for the T12HR input signal. 00 The input pin for T12HR _I0 01 The input pin for T12HR_I1 10 The input pin for T12HR_I2 11 The input pin for T12HR_I3
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

PISEL2
Port Input Select Register 2 **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
IST13HR	[1:0]	rw	**Input Select for T13HR** This bit field defines the port pin, that is used for the T13HR input signal. 00 The input pin for T13HR_I0 01 The input pin for T13HR_I1 10 The input pin for T13HR_I2 11 The input pin for T13HR_I3
0	[31:2]	r	**Reserved;** read as 0; should be written with 0.

Register CMPSTAT contains status bits monitoring the current capture and compare state and control bits defining the active/passive state of the compare channels.

CMPSTAT
Compare State Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T13IM	COUT63PS	COUT62PS	CC62PS	COUT61PS	CC61PS	COUT60PS	CC60PS	0	CC63ST	CCPOS2	CCPOS1	CCPOS0	CC62ST	CC61ST	CC60ST
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	r	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
CC60ST **CC61ST** **CC62ST** **CC63ST** 1)	0 1 2 6	rh	**Capture/Compare State Bits** Bits CC6xST monitor the state of the capture/compare channels. Bits CC6xST (x = 0, 1, 2) are related to T12, bit CC63ST is related to T13. 0 In compare mode, the timer count is less than the compare value. In capture mode, the selected edge has not yet been detected since the bit has been reset by software the last time. 1 In compare mode, the counter value is greater than or equal to the compare value. In capture mode, the selected edge has been detected.
CCPOS0 **CCPOS1** **CCPOS2**	3 4 5	rh	**Sampled Hall Pattern Bits** Bits CCPSOx (x = 0, 1, 2) are indicating the value of the input Hall pattern that has been compared to the current and expected value. The value is sampled when the event hcrdy (Hall compare ready) occurs. 0 The input CCPOSx has been sampled as 0. 1 The input CCPOSx has been sampled as 1.

Field	Bits	Type	Description
CC60PS **CC61PS** **CC62PS** **COUT60PS** **COUT61PS** **COUT62PS** **COUT63PS** **2)**	8 10 12 9 11 13 14	rwh	**Passive State Select for Compare Outputs** Bits CC6xPS, COUT6xPS select the state of the corresponding compare channel, which is considered to be the passive state. During the passive state, the passive level (defined in register PSLR) is driven by the output pin. Bits CC6xPS, COUT6xPS (x = 0, 1, 2) are related to T12, bit CC63PS is related to T13. 0 The corresponding compare output drives passive level while CC6xST is 0. 1 The corresponding compare output drives passive level while CC6xST is 1. In capture mode, these bits are not used.
T13IM[3)]	15	rwh	**T13 Inverted Modulation** Bit T13IM inverts the T13 signal for the modulation of the CC6x and COUT6x (x = 0, 1, 2) signals. 0 T13 output is not inverted. 1 T13 output is inverted for further modulation.
0	7, [31:16]	r	**Reserved;** read as 0; should be written with 0.

1) These bits are set and reset according to the T12, T13 switching rules.

2) These bits have shadow bits and are updated in parallel to the capture/compare registers of T12, T13 respectively. A read action targets the actually used values, whereas a write action targets the shadow bits.

3) This bit has a shadow bit and is updated in parallel to the compare and period registers of T13. A read action targets the actually used values, whereas a write action targets the shadow bit.

Register CMPMODIF contains control bits allowing for modification by software of the Capture/Compare state bits.

CMPMODIF
Compare State Modification Register **Reset Value: 0000 0000$_H$**

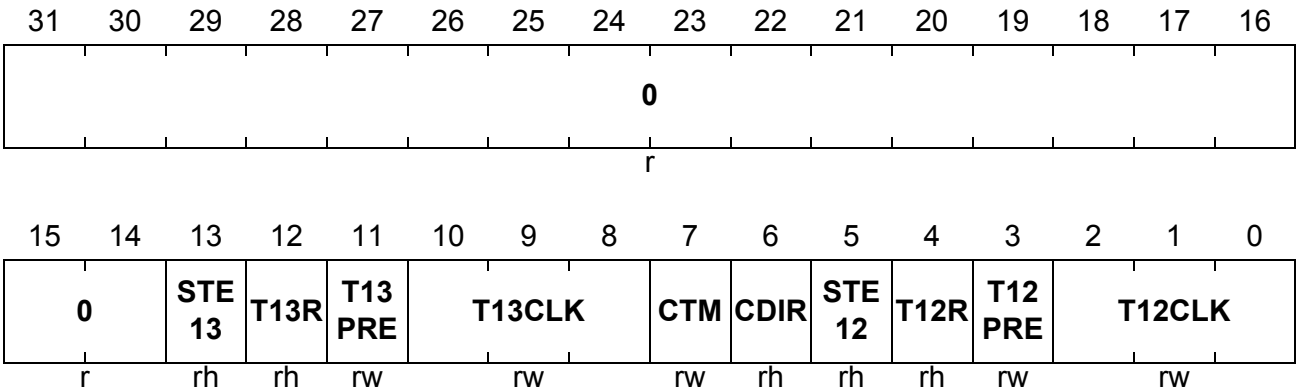
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	MCC 63R	0			MCC 62R	MCC 61R	MCC 60R	0	MCC 63S	0			MCC 62S	MCC 61S	MCC 60S
r	w	r			w	w	w	r	w	r			w	w	w

Field	Bits	Type	Description
MCC60S **MCC61S** **MCC62S** **MCC63S** **MCC60R** **MCC61R** **MCC62R** **MCC63R**	0 1 2 6 8 9 10 14	w	**Capture/Compare Status Modification Bits** These bits are used to bits to set (MCC6xS) or to reset (MCC6xR) the corresponding bits CC6xST by software. This feature allows the user to individually change the status of the output lines by software, e.g. when the corresponding compare timer is stopped. This allows a bit manipulation of CC6xST-bits by a single data write action. The following functionality of a write access to bits concerning the same capture/compare state bit is provided: MCC6xR, MCC6xS = 0, 0 Bit CC6xST is not changed 0, 1 Bit CC6xST is set 1, 0 Bit CC6xST is reset 1, 1 Reserved (toggle)
0	[5:3], 7, [13:11], [31:15]	r	**Reserved;** read as 0; should be written with 0.

Register TCTR0 controls the basic functionality of both timers T12 and T13.

TCTR0
Timer Control Register 0 **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
T12CLK	[2:0]	rw	**Timer T12 Input Clock Select** Selects the input clock for timer T12 which is derived from the peripheral clock according to the equation $f_{T12} = f_{CCU} / 2^{<T12CLK>}$. 000 $f_{T12} = f_{CCU}$ 001 $f_{T12} = f_{CCU} / 2$ 010 $f_{T12} = f_{CCU} / 4$ 011 $f_{T12} = f_{CCU} / 8$ 100 $f_{T12} = f_{CCU} / 16$ 101 $f_{T12} = f_{CCU} / 32$ 110 $f_{T12} = f_{CCU} / 64$ 111 $f_{T12} = f_{CCU} / 128$
T12PRE	3	rw	**Timer T12 Prescaler Bit** In order to support higher clock frequencies, an additional prescaler factor of 1/256 can be enabled for the prescaler for T12. 0 The additional prescaler for T12 is disabled 1 The additional prescaler for T12 is enabled
T12R[1)]	4	rh	**Timer T12 Run Bit** T12R starts and stops timer T12. It is set/reset by software by setting bits T12RR orT12RS or it is reset by hardware according to the function defined by bit field T12SSC. 0 Timer T12 is stopped 1 Timer T12 is running

Field	Bits	Type	Description
STE12	5	rh	**Timer T12 Shadow Transfer Enable** Bit STE12 enables or disables the shadow transfer of the T12 period value, the compare values and passive state select bits and levels from their shadow registers to the actual registers if a T12 shadow transfer event is detected. Bit STE12 is cleared by hardware after the shadow transfer. A T12 shadow transfer event is a period-match while counting up or a one-match while counting down. 0 The shadow register transfer is disabled 1 The shadow register transfer is enabled
CDIR	6	rh	**Count Direction of Timer T12** This bit is set/reset according to the counting rules of T12. 0 T12 counts up 1 T12 counts down
CTM	7	rw	**T12 Operating Mode** 0 Edge-aligned Mode: T12 always counts up and continues counting from zero after reaching the period value. 1 Center-aligned Mode: T12 counts down after detecting a period-match and counts up after detecting a one-match.
T13CLK	[10:8]	rw	**Timer T13 Input Clock Select** Selects the input clock for timer T13 which is derived from the peripheral clock according to the equation $f_{T13} = f_{CCU} / 2^{<T13CLK>}$. 000 $f_{T13} = f_{CCU}$ 001 $f_{T13} = f_{CCU} / 2$ 010 $f_{T13} = f_{CCU} / 4$ 011 $f_{T13} = f_{CCU} / 8$ 100 $f_{T13} = f_{CCU} / 16$ 101 $f_{T13} = f_{CCU} / 32$ 110 $f_{T13} = f_{CCU} / 64$ 111 $f_{T13} = f_{CCU} / 128$

Field	Bits	Type	Description
T13PRE	11	rw	**Timer T13 Prescaler Bit** In order to support higher clock frequencies, an additional prescaler factor of 1/256 can be enabled for the prescaler for T13. 0 The additional prescaler for T13 is disabled 1 The additional prescaler for T13 is enabled
T13R[2)]	12	rh	**Timer T13 Run Bit** T13R starts and stops timer T13. It is set/reset by software by setting bits T13RR orT13RS or it is set/reset by hardware according to the function defined by bit fields T13SSC, T13TEC and T13TED. 0 Timer T13 is stopped 1 Timer T13 is running
STE13	13	rh	**Timer T13 Shadow Transfer Enable** Bit STE13 enables or disables the shadow transfer of the T13 period value, the compare value and passive state select bit and level from their shadow registers to the actual registers if a T13 shadow transfer event is detected. Bit STE13 is cleared by hardware after the shadow transfer. A T13 shadow transfer event is a period-match. 0 The shadow register transfer is disabled 1 The shadow register transfer is enabled
0	[31:14]	r	**Reserved;** read as 0; should be written with 0.

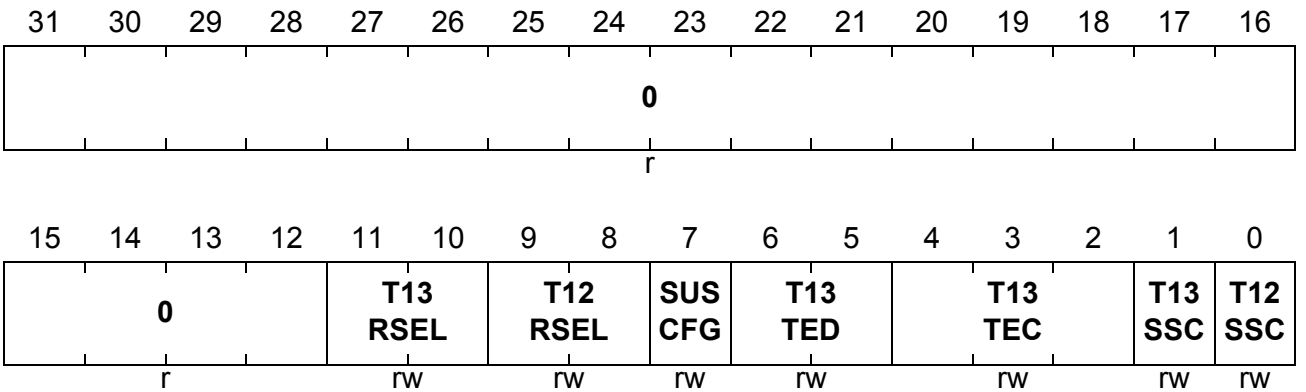
1) A concurrent set/reset action on T12R (from T12SSC, T12RR or T12RS) will have no effect. The bit T12R will remain unchanged.

2) A concurrent set/reset action on T13R (from T13SSC, T13TEC, T13RR or T13RS) will have no effect. The bit T12R will remain unchanged.

Note: A write action to the bit fields T12CLK or T12PRE is only taken into account while the timer T12 is not running (T12R = 0). A write action to the bit fields T13CLK or T13PRE is only taken into account while the timer T13 is not running (T13R = 0).

Register TCTR2 controls the single-shot and the synchronization functionality of both timers T12 and T13. Both timers can run in single-shot mode. In this mode they stop their counting sequence automatically after one counting period with a count value of zero. The single-shot mode and the synchronization feature of T13 to T12 allow the generation of events with a programmable delay after well-defined PWM actions of T12. For example, this feature can be used to trigger AD conversions after a specified delay (to avoid problems due to switching noise) synchronously to a PWM event.

TCTR2
Timer Control Register 2 **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
T12SSC	0	rw	**Timer T12 Single Shot Control** This bit controls the single shot-mode of T12. 0 The single-shot mode is disabled, no hardware action on T12R. 1 The single shot mode is enabled, the bit T12R is reset by hardware if - T12 reaches its period value in edge-aligned mode - T12 reaches the value 1 while down counting in center-aligned mode. In parallel to the reset action of bit T12R, the bits CC6xST (x = 0, 1, 2) are reset.
T13SSC	1	rw	**Timer T13 Single Shot Control** This bit controls the single shot-mode of T13. 0 No hardware action on T13R 1 The single-shot mode is enabled, the bit T13R is reset by hardware if T13 reaches its period value. In parallel to the reset action of bit T13R, the bit CC63ST is reset.

Field	Bits	Type	Description
T13TEC	[4:2]	rw	**T13 Trigger Event Control** Bit field T13TEC selects the trigger event to start T13 (automatic set of T13R for synchronization to T12 compare signals) according to following combinations: 000 No action 001 Set T13R on a T12 compare event on channel 0 010 Set T13R on a T12 compare event on channel 1 011 Set T13R on a T12 compare event on channel 2 100 Set T13R on any T12 compare event on the channels 0, 1, or 2 101 Set T13R upon a period-match of T12 110 Set T13R upon a zero-match of T12 (while counting up) 111 Set T13R on any edge of inputs CCPOSx
T13TED[1)]	[6:5]	rw	**Timer T13 Trigger Event Direction** Bit field T13TED delivers additional information to control the automatic set of bit T13R in the case that the trigger action defined by T13TEC is detected. 00 Reserved, no action 01 While T12 is counting up 10 While T12 is counting down 11 Independent on the count direction of T12
SUSCFG	7	rw	**Suspend Configuration** Bit SUSCFG defines the behavior of the module while the suspend request is active (independent of the status of the acknowledge signal). In any case, the timers T12 and T13 are stopped when reaching the end of their period. 0 No additional action 1 The outputs enabled for trap functionality are set to their passive values, see **Figure 29-23** and **Figure 29-24**.

Field	Bits	Type	Description
T12RSEL	[9:8]	rw	**Timer T12 External Run Selection** Bit field T12RSEL defines the event of signal T12HR that can set the run bit T12R by hardware. 00 The external setting of T12R is disabled 01 Bit T12R is set if a rising edge of signal T12HR is detected 10 Bit T12R is set if a falling edge of signal T12HR is detected 11 Bit T12R is set if an edge of signal T12HR is detected
T13RSEL	[11:10]	rw	**Timer T13 External Run Selection** Bit field T13RSEL defines the event of signal T13HR that can set the run bit T13R by hardware. 00 The external setting of T13R is disabled 01 Bit T13R is set if a rising edge of signal T13HR is detected 10 Bit T13R is set if a falling edge of signal T13HR is detected 11 Bit T13R is set if an edge of signal T13HR is detected
0	[31:12]	r	**Reserved;** read as 0; should be written with 0.

1) Example:
If the timer T13 is intended to start at any compare event on T12 (T13TEC = 100_B) the trigger event direction can be programmed to
- counting up >> a T12 channel 0, 1, 2 compare match triggers T13R only while T12 is counting up
- counting down >> a T12 channel 0, 1, 2 compare match triggers T13R only while T12 is counting down
- independent from bit CDIR >> each T12 channel 0, 1, 2 compare match triggers T13R

The timer count direction is taken from the value of bit CDIR. As a result, if T12 is running in edge-aligned mode (counting up only), T13 can only be started automatically if bit field T13TED = 01_B or 11_B.

Register TCTR4 allows the software control of the run bits T12R and T13R by independent set and reset conditions. Furthermore, the timers can be reset (while running) and the bits STE12 and STE13 can be controlled by software.

TCTR4
Timer Control Register 4 **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T13 STD	T13 STR	0			T13 RES	T13 RS	T13 RR	T12 STD	T12 STR	0		DT RES	T12 RES	T12 RS	T12 RR
w	w	r			w	w	w	w	w	r		w	w	w	w

Field	Bits	Type	Description
T12RR	0	w	**Timer T12 Run Reset** Setting this bit resets the T12R bit. 0 T12R is not influenced 1 T12R is cleared, T12 stops counting
T12RS	1	w	**Timer T12 Run Set** Setting this bit sets the T12R bit. 0 T12R is not influenced 1 T12R is set, T12 counts
T12RES	2	w	**Timer T12 Reset** 0 No effect on T12 1 The T12 counter register is reset to zero. The switching of the output signals is according to the switching rules. Setting of T12RES has no impact on bit T12R.
DTRES	3	w	**Dead-Time Counter Reset** 0 No effect on the dead-time counters 1 The three dead-time counter channels are reset to zero
T12STR	6	w	**Timer T12 Shadow Transfer Request** 0 No action 1 STE12 is set, enabling the shadow transfer

Field	Bits	Type	Description
T12STD	7	w	**Timer T12 Shadow Transfer Disable** 0 No action 1 STE12 is reset without triggering the shadow transfer
T13RR	8	w	**Timer T13 Run Reset** Setting this bit resets the T13R bit. 0 T13R is not influenced 1 T13R is cleared, T13 stops counting
T13RS	9	w	**Timer T13 Run Set** Setting this bit sets the T13R bit. 0 T13R is not influenced 1 T13R is set, T13 counts
T13RES	10	w	**Timer T13 Reset** 0 No effect on T13 1 The T13 counter register is reset to zero. The switching of the output signals is according to the switching rules. Setting of T13RES has no impact on bit T13R.
T13STR	14	w	**Timer T13 Shadow Transfer Request** 0 No action 1 STE13 is set, enabling the shadow transfer
T13STD	15	w	**Timer T13 Shadow Transfer Disable** 0 No action 1 STE13 is reset without triggering the shadow transfer
0	[5:4], [13:11], [31:16]	r	**Reserved;** read as 0; should be written with 0.

Note: A simultaneous write of a 1 to bits which set and reset the same bit will trigger no action. The corresponding bit will remain unchanged.

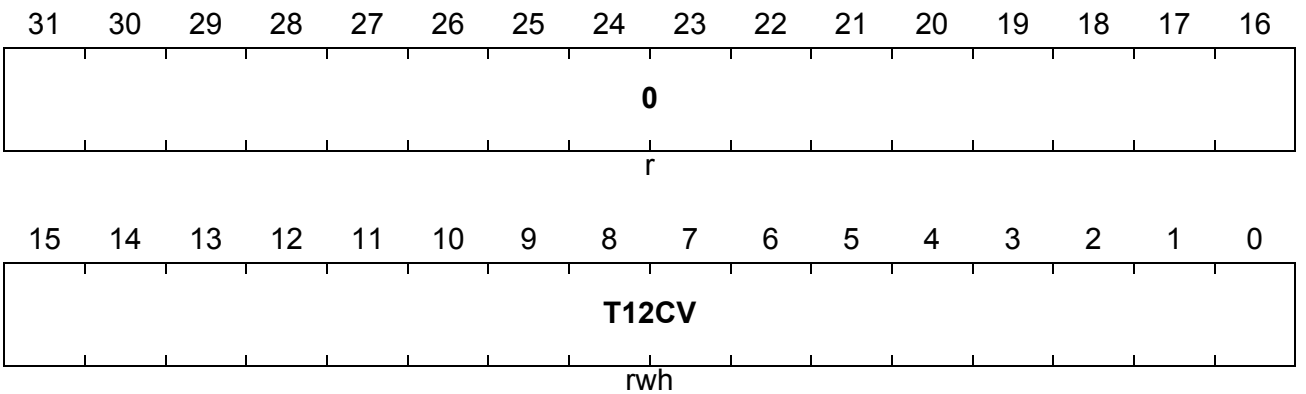
29.2.2 Timer12 - Related Registers

The generation of the patterns for a 3-channel pulse width modulation (PWM) is based on timer T12. The registers related to timer T12 can be concurrently updated (with well-defined conditions) in order to ensure consistency of the three PWM channels.
Timer T12 supports capture and compare modes, which can be independently selected for the three channels CC60, CC61 and CC62.

Register T12 represents the counting value of timer T12. It can only be written while the timer T12 is stopped. Write actions while T12 is running are not taken into account. Register T12 can always be read by software.

In edge-aligned mode, T12 only counts up, whereas in center-aligned mode, T12 can count up and down.

T12
Timer T12 Counter Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
T12CV	[15:0]	rwh	**Timer 12 Counter Value** This register represents the 16-bit counter value of Timer12.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

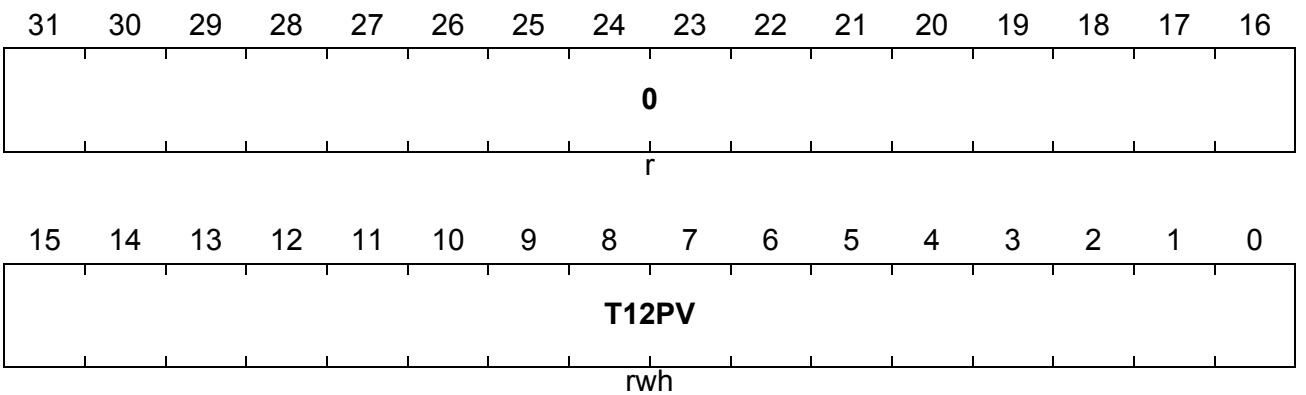
Note: While timer T12 is stopped, the internal clock divider is reset in order to ensure reproducible timings and delays.

Note: The timer period, compare values, passive state selects bits and passive levels bits for both timers are written to shadow registers and not directly to the actual registers. Thus, the values for a new output signal can be programmed without disturbing the currently generated signal(s). The transfer from the shadow registers to the actual registers is enabled by setting the respective shadow transfer enable bit STEx.
If the transfer is enabled, the shadow registers are copied to the respective

registers as soon as the associated timer reaches the value zero the next time (being cleared in edge-aligned mode or counting down from 1 in center-aligned mode). When timer T12 is operating in center-aligned mode, it will also copy the registers (if enabled by STE12) if it reaches the currently programmed period value (counting up).
When a timer is stopped (TxR = 0), the shadow transfer takes place immediately if the corresponding bit STEx is set.
After the transfer, the respective bit STEx is cleared automatically.

Register T12PR contains the period value for timer T12. The period value is compared to the actual counter value of T12 and the resulting counter actions depend on the defined counting rules. This register has a shadow register and the shadow transfer is controlled by bit STE12. A read action by software delivers the value which is currently used for the compare action, whereas the write action targets a shadow register. The shadow register structure allows a concurrent update of all T12-related values.

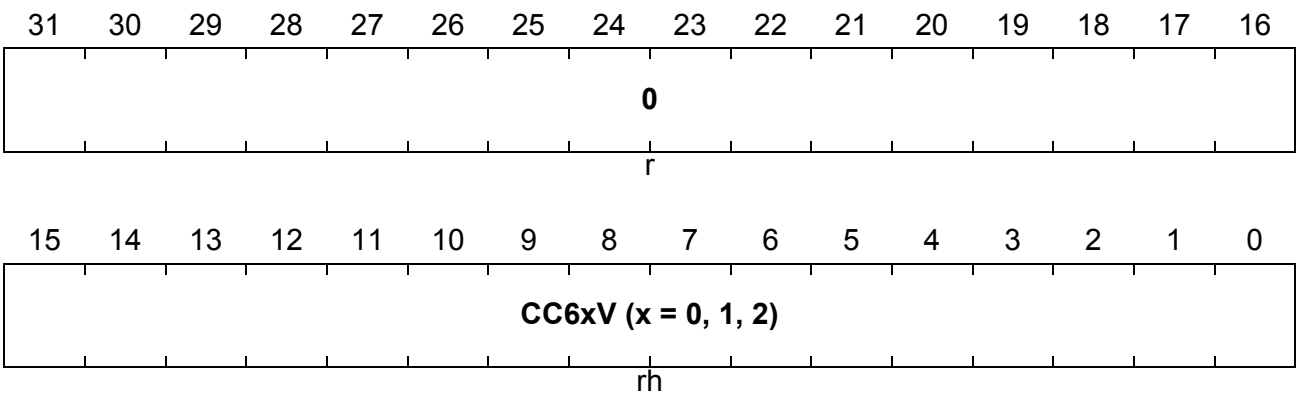
T12PR
Timer T12 Period Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
T12PV	[15:0]	rwh	**T12 Period Value** The value T12PV defines the counter value for T12, which leads to a period-match. When reaching this value, the timerT12 is set to zero (edge-aligned mode) or changes its count direction to down counting (center-aligned mode).
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

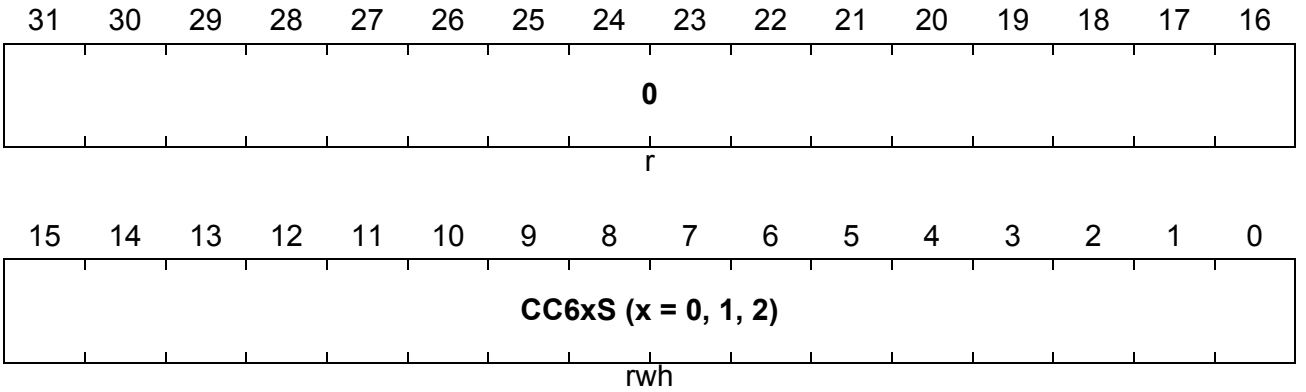
In compare mode, the registers CC6xR (x = 0, 1, 2) are the actual compare registers for T12. The values stored in CC6xR are compared (all three channels in parallel) to the counter value of T12. In capture mode, the current value of the T12 counter register is captured by registers CC6xR if the corresponding capture event is detected.
The registers CC6xR can only be read by software, the modification of the value is done by a shadow register transfer from register CC6xSR. The corresponding shadow registers CC6xSR can be read and written by software. In capture mode, the value of the T12 counter register can also be captured by registers CC6xSR if the selected capture event is detected (depending on the selected mode).

CC6xR (x = 0, 1, 2)
Capture/Compare Register for Channel CC6x **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CC6xV (x = 0, 1, 2)	[15:0]	rh	**Channel x Capture/Compare Value** In compare mode, the bit fields CC6xV contain the values, that are compared to the T12 counter value. In capture mode, the captured value of T12 can be read from these registers.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

CC6xSR (x = 0, 1, 2)
Capture/Compare Shadow Register for Channel CC6x **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CC6xS (x = 0, 1, 2)	[15:0]	rwh	**Shadow Register for Channel x Capture/Compare Value** In compare mode, the bit fields contents of CC6xS are transferred to the bit fields CC6xV during a shadow transfer. In capture mode, the captured value of T12 can be read from these registers.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

Register T12DTC controls the dead-time generation for the timer T12 compare channels. Each channel can be independently enabled/disabled for dead-time generation. If enabled, the transition from passive state to active state is delayed by the value defined by bit field DTM. The dead-time counter can only be reloaded while it is zero.

T12DTC
Dead-Time Control Register for Timer12 **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	DTR2	DTR1	DTR0	0	DTE2	DTE1	DTE0	DTM							
r	rh	rh	rh	r	rw	rw	rw	rw							

Field	Bits	Type	Description
DTM	[7:0]	rw	**Dead-Time** Bit field DTM determines the programmable delay between switching from the passive state to the active state of the selected outputs. The switching from the active state to the passive state is not delayed.
DTE2 **DTE1** **DTE0**	10 9 8	rw	**Dead Time Enable Bits** Bits DTE0 … DTE2 enable and disable the dead time generation for each compare channel (0, 1, 2) of timer T12. 0 Dead time generation is disabled. The corresponding outputs switch from the passive state to the active state (according to the actual compare status) without any delay. 1 Dead time generation is enabled. The corresponding outputs switch from the passive state to the active state (according to the compare status) with the delay programmed in bit field DTM.

Field	Bits	Type	Description
DTR2 **DTR1** **DTR0**	14 13 12	rh	**Dead Time Run Indication Bits** Bits DTR0 … DTR2 indicate the status of the dead time generation for each compare channel (0, 1, 2) of timer T12. 0 The value of the corresponding dead time counter channel is 0. 1 The value of the corresponding dead time counter channel is not 0.
0	11, [31:15]	r	**Reserved;** read as 0; should be written with 0.

Note: The dead time counters are clocked with the same frequency as T12. This structure allows symmetrical dead time generation in center-aligned and in edge-aligned PWM mode. A duty cycle of 50% leads to CC6x, COUT6x switched on for: 0.5 × period - dead-time.

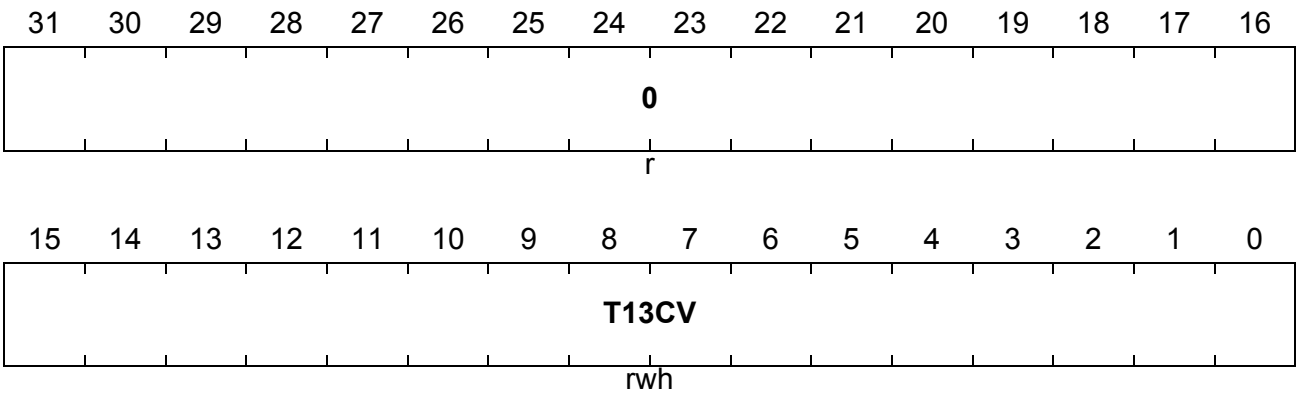
Note: The dead-time counters are not reset by bit T12RES, but by bit DTRES.

29.2.3 Timer13 - Related Registers

The generation of the patterns for a single channel pulse width modulation (PWM) is based on timer T13. The registers related to timer T13 can be concurrently updated (with well-defined conditions) in order to ensure consistency of the PWM signal. T13 can be synchronized to several timer T12 events.

Timer T13 only supports compare mode on its compare channel CC63. Register T13 represents the counting value of timer T13. It can only be written while the timer T13 is stopped. Write actions while T13 is running are not taken into account. Register T13 can always be read by software. Timer T13 only supports edge-aligned mode (counting up).

T13
Timer T13 Register **Reset Value: 0000 0000$_H$**

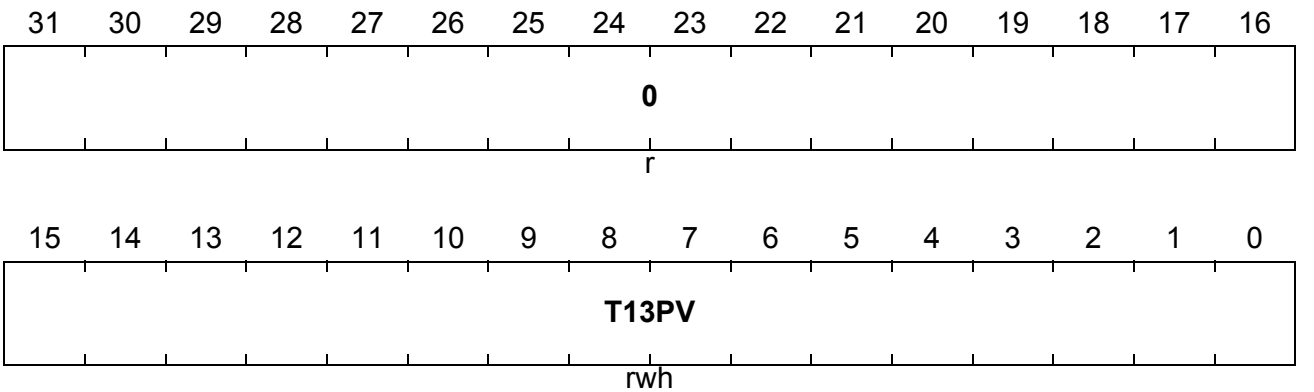


Field	Bits	Type	Description
T13CV	[15:0]	rwh	**Timer 13 Counter Value** This register represents the 16-bit counter value of Timer 13.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

Note: While timer T13 is stopped, the internal clock divider is reset in order to ensure reproducible timings and delays.

Register T13PR contains the period value for timer T13. The period value is compared to the actual counter value of T13 and the resulting counter actions depend on the defined counting rules. This register has a shadow register and the shadow transfer is controlled by bit STE13. A read action by software delivers the value which is currently used for the compare action, whereas the write action targets a shadow register. The shadow register structure allows a concurrent update of all T13-related values.

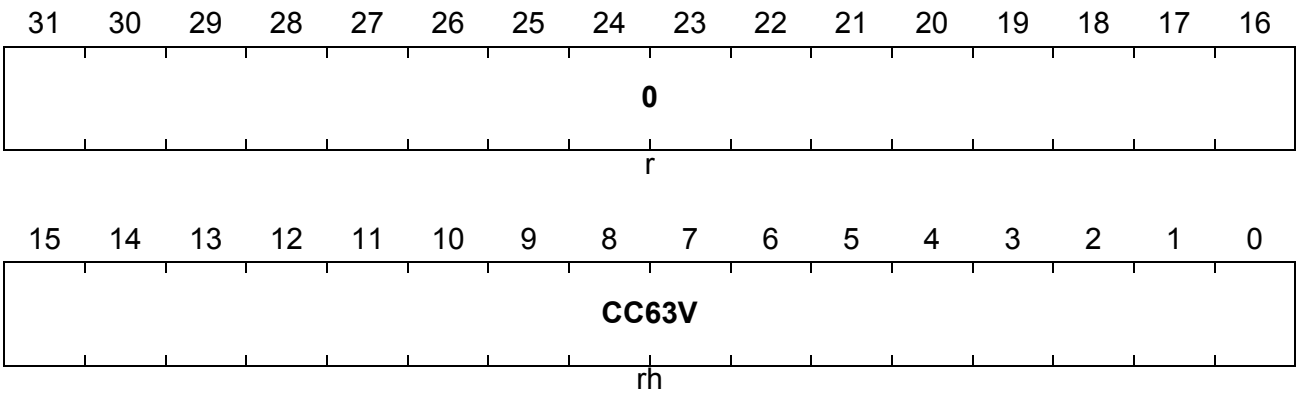
T13PR
Timer T13 Period Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
T13PV	[15:0]	rwh	**T13 Period Value** The value T13PV defines the counter value for T13, which leads to a period-match. When reaching this value, the timer T13 is set to zero.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

Register CC63R is the actual compare register for T13. The value stored in CC63R is compared to the counter value of T13. The register CC63R can only be read by software, the modification of the value is done by a shadow register transfer from register CC63SR. The corresponding shadow register CC63SR can be read and written by software.

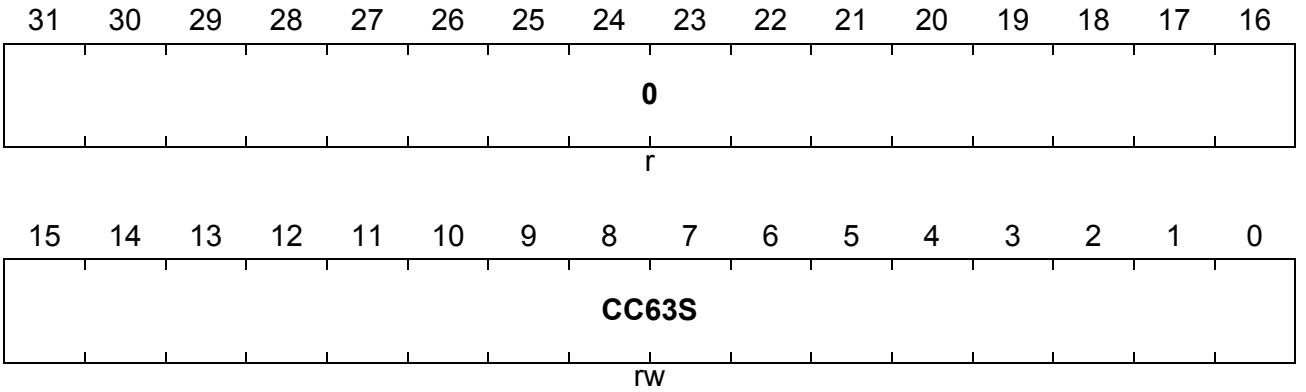
CC63R
Compare Register for Channel CC63 **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CC63V	[15:0]	rh	**Channel CC63 Compare Value** The bit field CC63V contains the value, that is compared to the T13 counter value.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

CC63SR
Compare Shadow Register for Channel CC63

Reset Value: 0000 0000$_H$

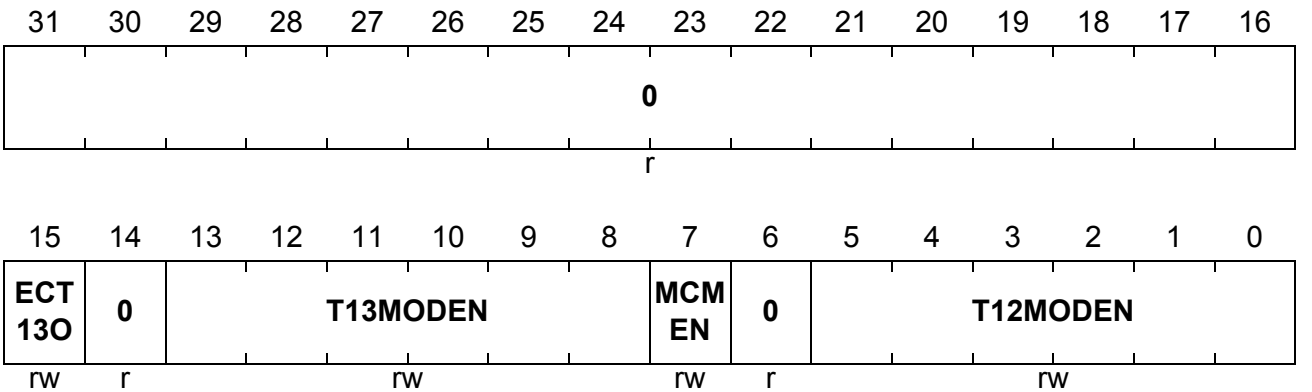


Field	Bits	Type	Description
CC63S	[15:0]	rw	**Shadow Register for Channel CC63 Compare Value** The bit field contents of CC63S is transferred to the bit field CC63V during a shadow transfer.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

29.2.4 Modulation Control Registers

Register MODCTR contains control bits enabling the modulation of the corresponding output signal by PWM pattern generated by the timers T12 and T13. Furthermore, the multi-channel mode can be enabled as additional modulation source for the output signals.

MODCTR
Modulation Control Register **Reset Value: 0000 0000$_H$**

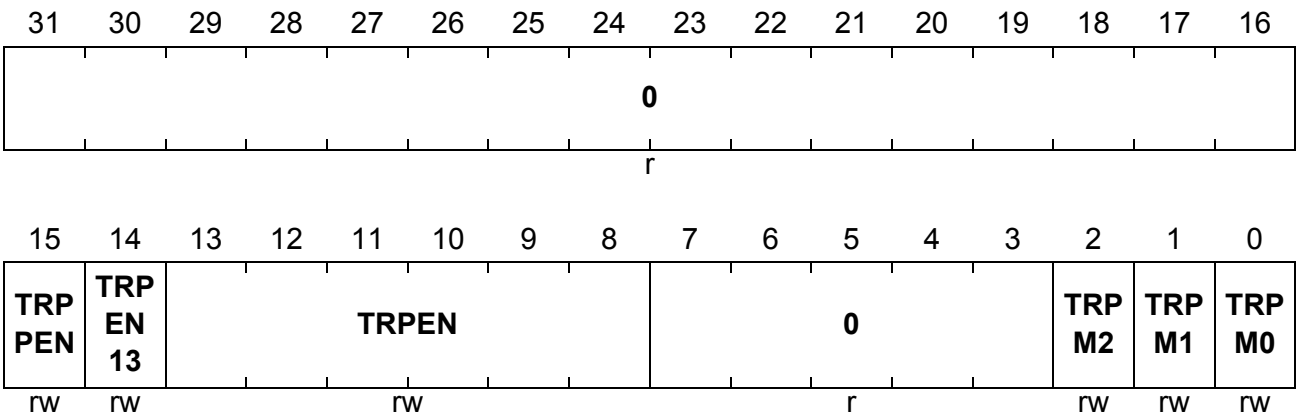


Field	Bits	Type	Description
T12MODEN	[5:0]	rw	**T12 Modulation Enable** Setting these bits enables the modulation of the corresponding compare channel by a PWM pattern generated by timer T12. The bit positions are corresponding to the following output signals: Bit 0 Modulation of CC60 Bit 1 Modulation of COUT60 Bit 2 Modulation of CC61 Bit 3 Modulation of COUT61 Bit 4 Modulation of CC62 Bit 5 Modulation of COUT62 The enable feature of the modulation is defined as follows: 0 The modulation of the corresponding output signal by a T12 PWM pattern is disabled. 1 The modulation of the corresponding output signal by a T12 PWM pattern is enabled.

Field	Bits	Type	Description
MCMEN	7	rw	**Multi-Channel Mode Enable** 0 The modulation of the corresponding output signal by a multi-channel pattern according to bit field MCMP is disabled. 1 The modulation of the corresponding output signal by a multi-channel pattern according to bit field MCMP is enabled.
T13MODEN	[13:8]	rw	**T13 Modulation Enable** Setting these bits enables the modulation of the corresponding compare channel by a PWM pattern generated by timer T13. The bit positions are corresponding to the following output signals: Bit 8 Modulation of CC60 Bit 9 Modulation of COUT60 Bit 10 Modulation of CC61 Bit 11 Modulation of COUT61 Bit 12 Modulation of CC62 Bit 13 Modulation of COUT62 The enable feature of the modulation is defined as follows: 0 The modulation of the corresponding output signal by a T13 PWM pattern is disabled. 1 The modulation of the corresponding output signal by a T13 PWM pattern is enabled.
ECT13O	15	rw	**Enable Compare Timer T13 Output** 0 The alternate output function COUT63 is disabled. 1 The alternate output function COUT63 is enabled for the PWM signal generated by T13.
0	6, 14, [31:16]	r	**Reserved;** read as 0; should be written with 0.

Register TRPCTR controls the trap functionality. It contains independent enable bits for each output signal and control bits to select the behavior in case of a trap condition. The trap condition is a low-level on the CTRAP input pin, which is monitored (inverted level) by bit TRPF (in register IS). While TRPF = 1 (trap input active), the trap state bit TRPS (in register IS) is set to 1.

TRPCTR
Trap Control Register **Reset Value: 0000 0000$_H$**



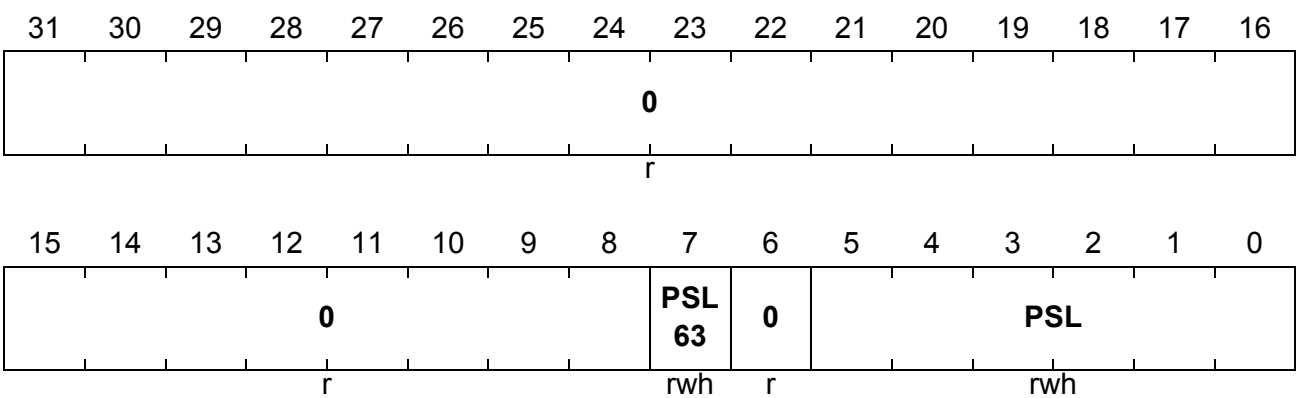
Field	Bits	Type	Description
TRPM1, TRPM0	[1:0]	rw	**Trap Mode Control Bits 1, 0** These two bits define the behavior of the selected outputs when leaving the trap state after the trap condition has become inactive again. A synchronization to the timer driving the PWM pattern permits to avoid unintended short pulses when leaving the trap state. The combination (TRPM1, TRPM0) leads to: 00 The trap state is left (return to normal operation according to TRPM2) when a zero-match of T12 (while counting up) is detected (synchronization to T12). 01 The trap state is left (return to normal operation according to TRPM2) when a zero-match of T13 is detected (synchronization to T13). 10 Reserved 11 The trap state is left (return to normal operation according to TRPM2) immediately without any synchronization to T12 or T13.

Field	Bits	Type	Description
TRPM2	2	rw	**Trap Mode Control Bit 2** 0 The trap state can be left (return to normal operation = bit TRPS = 0) as soon as the input $\overline{\text{CTRAP}}$ becomes inactive. Bit TRPF is automatically cleared by hardware if the input pin $\overline{\text{CTRAP}}$ becomes 1. Bit TRPS is automatically cleared by hardware if bit TRPF is 0 and if the synchronization condition (according to TRPM0, 1) is detected. 1 The trap state can be left (return to normal operation = bit TRPS = 0) as soon as bit TRPF is reset by software after the input $\overline{\text{CTRAP}}$ becomes inactive (TRPF is not cleared by hardware). Bit TRPS is automatically cleared by hardware if bit TRPF = 0 and if the synchronization condition (according to TRPM0, 1) is detected.
TRPEN	[13:8]	rw	**Trap Enable Control** Setting these bits enables the trap functionality for the following corresponding output signals: Bit 8 Trap functionality of CC60 Bit 9 Trap functionality of COUT60 Bit 10 Trap functionality of CC61 Bit 11 Trap functionality of COUT61 Bit 12 Trap functionality of CC62 Bit 13 Trap functionality of COUT62 The enable feature of the trap functionality is defined as follows: 0 The trap functionality of the corresponding output signal is disabled. The output state is independent from bit TRPS. 1 The trap functionality of the corresponding output signal is enabled. The output is set to the passive state while TRPS = 1.

Field	Bits	Type	Description
TRPEN13	14	rw	**Trap Enable Control for Timer T13** 0 The trap functionality for T13 is disabled. Timer T13 (if selected and enabled) provides PWM functionality even while TRPS = 1. 1 The trap functionality for T13 is enabled. The timer T13 PWM output signal is set to the passive state while TRPS = 1.
TRPPEN	15	rw	**Trap Pin Enable** 0 The trap functionality based on the input pin CTRAP is disabled. A trap can only be generated by software by setting bit TRPF. 1 The trap functionality based on the input pin CTRAP is enabled. A trap can be generated by software by setting bit TRPF or by CTRAP = 0.
0	[7:3], [31:16]	r	**Reserved;** read as 0; should be written with 0.

Register PSLR defines the passive state level driven by the output pins of the module. The passive state level is the value that is driven by the port pin during the passive state of the output. During the active state, the corresponding output pin drives the active state level, which is the inverted passive state level. The passive state level permits to adapt the driven output levels to the driver polarity (inverted, not inverted) of the connected power stage.

PSLR
Passive State Level Register **Reset Value: 0000 0000$_H$**



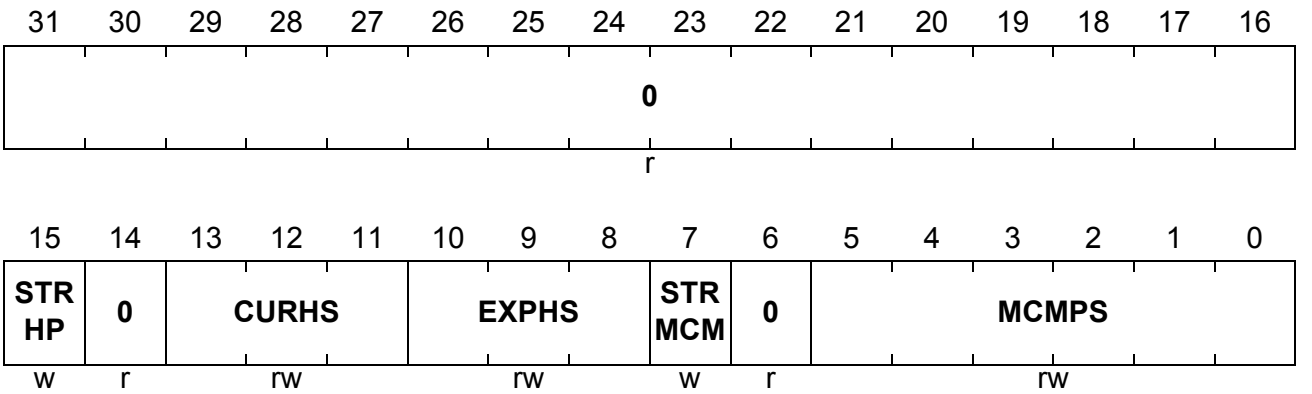
Field	Bits	Type	Description
PSL[1)]	[5:0]	rwh	**Compare Outputs Passive State Level** The bits of this bit field define the passive level driven by the module outputs during the passive state. The bit positions are: Bit 0 Passive level for output CC60 Bit 1 Passive level for output COUT60 Bit 2 Passive level for output CC61 Bit 3 Passive level for output COUT61 Bit 4 Passive level for output CC62 Bit 5 Passive level for output COUT62 The value of each bit position is defined as: 0 The passive level is 0. 1 The passive level is 1.
PSL63[2)]	7	rwh	**Passive State Level of Output COUT63** This bit field defines the passive level of the output pin COUT63. 0 The passive level is 0. 1 The passive level is 1.
0	6, [31:8]	r	**Reserved;** read as 0; should be written with 0.

1) Bit field PSL has a shadow registers to allow for updates without undesired pulses on the output lines. The bits are updated with the T12 shadow transfer. A read action targets the actually used values, whereas a write action targets the shadow bits.

2) Bit PSL63 has a shadow register to allow for updates without undesired pulses on the output line. The bit is updated with the T13 shadow transfer. A read action targets the actually used values, whereas a write action targets the shadow bits.

Register MCMOUTS contains bits controlling the output states for multi-channel mode. Furthermore, the appropriate signals for the block commutation by Hall sensors can be selected. This register is a shadow register (that can be written) for register MCMOUT, which indicates the currently active signals.

MCMOUTS
Multi-Channel Mode Output Shadow Register **Reset Value: 0000 0000$_H$**

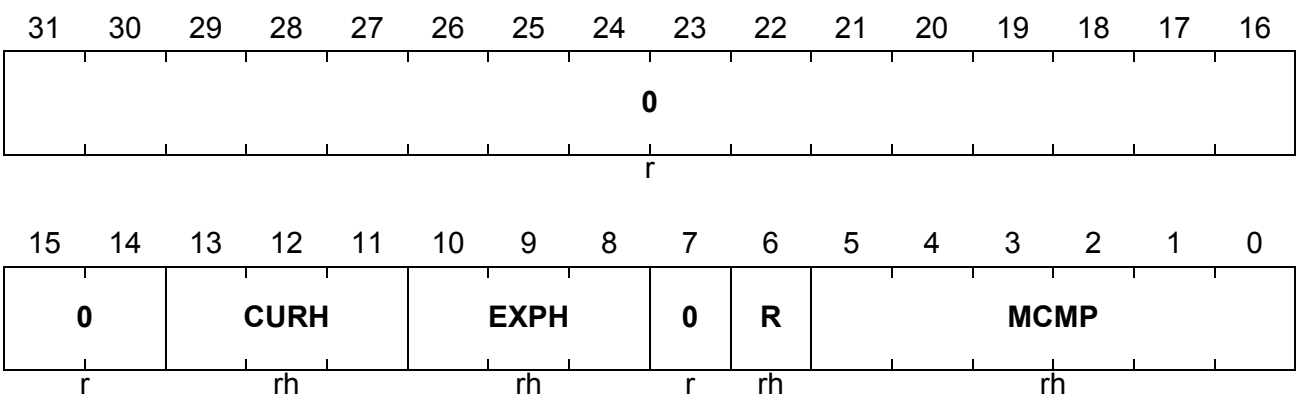


Field	Bits	Type	Description
MCMPS	[5:0]	rw	**Multi-Channel PWM Pattern Shadow** Bit field MCMPS is the shadow bit field for bit field MCMP. The multi-channel shadow transfer is triggered according to the transfer conditions defined by register MCMCTR.
STRMCM	7	w	**Shadow Transfer Request for MCMPS** Setting this bits during a write action leads to an immediate update of bit field MCMP by the value written to bit field MCMPS. This functionality permits an update triggered by software. When read, this bit always delivers 0. 0 Bit field MCMP is updated according to the defined hardware action. The write access to bit field MCMPS does not modify bit field MCMP. 1 Bit field MCMP is updated by the value written to bit field MCMPS.
EXPHS	[10:8]	rw	**Expected Hall Pattern Shadow** Bit field EXPHS is the shadow bit field for bit field EXPH. The bit field is transferred to bit field EXPH if an edge on the hall input pins CCPOSx (x = 0, 1, 2) is detected.

Field	Bits	Type	Description
CURHS	[11:13]	rw	**Current Hall Pattern Shadow** Bit field CURHS is the shadow bit field for bit field CURH. The bit field is transferred to bit field CURH if an edge on the hall input pins CCPOSx (x = 0, 1, 2) is detected.
STRHP	15	w	**Shadow Transfer Request for the Hall Pattern** Setting this bits during a write action leads to an immediate update of bit fields CURH and EXPH by the value written to bit fields CURHS and EXPH. This functionality permits an update triggered by software. When read, this bit always delivers 0. 0 The bit fields CURH and EXPH are updated according to the defined hardware action. The write access to bit fields CURHS and EXPHS does not modify the bit fields CURH and EXPH. 1 The bit fields CURH and EXPH are updated by the value written to the bit fields CURHS and EXPHS.
0	6, 14, [31:16]	r	**Reserved;** read as 0; should be written with 0.

Register MCMOUT shows the multi-channel control bits that are currently used. Register MCMOUT is defined as follows:

MCMOUT
Multi-Channel Mode Output Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
MCMP[1)]	[5:0]	rh	**Multi-Channel PWM Pattern** Bit field MCMP is written by a shadow transfer from bit field MCMPS. It contains the output pattern for the multi-channel mode. If this mode is enabled by bit MCMEN in register MODCTR, the output state of the following output signal can be modified: Bit 0 Multi-channel state for output CC60 Bit 1 Multi-channel state for output COUT60 Bit 2 Multi-channel state for output CC61 Bit 3 Multi-channel state for output COUT61 Bit 4 Multi-channel state for output CC62 Bit 5 Multi-channel state for output COUT62 The multi-channel patterns can set the related output to the passive state. 0 The output is set to the passive state. The PWM generated by T12 or T13 are not taken into account. 1 The output can deliver the PWM generated by T12 or T13 (according to register MODCTR).
R	6	rh	**Reminder Flag** This reminder flag indicates that the shadow transfer from bit field MCMPS to MCMP has been requested by the selected trigger source. This bit is cleared when the shadow transfer takes place and while MCMEN = 0. 0 Currently, no shadow transfer from MCMPS to MCMP is requested. 1 A shadow transfer from MCMPS to MCMP has been requested by the selected trigger source, but it has not yet been executed, because the selected synchronization condition has not yet occurred.

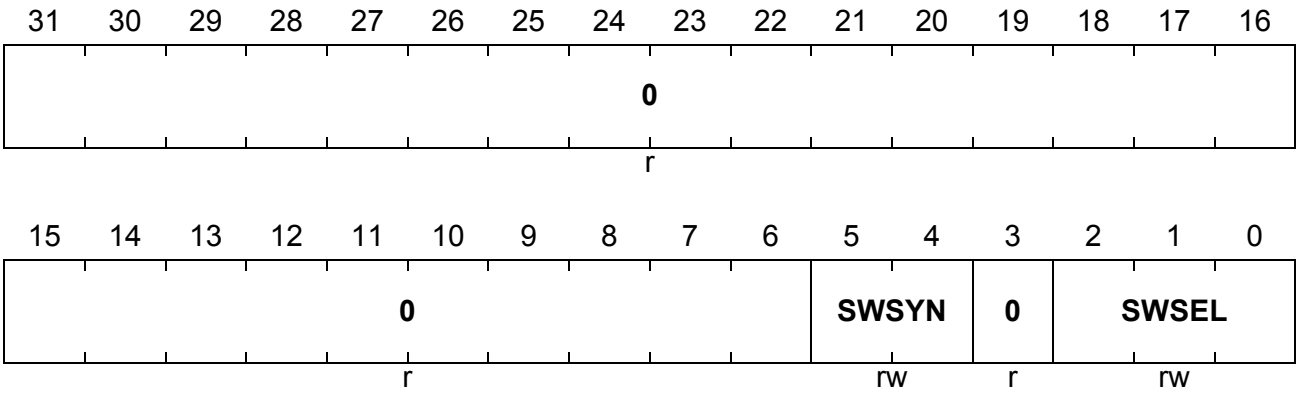
Field	Bits	Type	Description
EXPH[2)]	[10:8]	rh	**Expected Hall Pattern** Bit field EXPH is written by a shadow transfer from bit field EXPHS. The contents are compared after every detected edge at the hall input pins with the pattern at the hall input pins in order to detect the occurrence of the next desired (= expected) hall pattern or a wrong pattern. If the current hall pattern at the hall input pins is equal to the bit field EXPH, bit CHE (correct hall event) is set and an interrupt request is generated (if enabled by bit ENCHE). If the current hall pattern at the hall input pins is not equal to the bit fields CURH or EXPH, bit WHE (wrong hall event) is set and an interrupt request is generated (if enabled by bit ENWHE).
CURH	[13:11]	rh	**Current Hall Pattern** Bit field CURH is written by a shadow transfer from bit field CURHS.The contents are compared after every detected edge at the hall input pins with the pattern at the hall input pins in order to detect the occurrence of the next desired (= expected) hall pattern or a wrong pattern. If the current hall input pattern is equal to bit field CURH, the detected edge at the hall input pins has been an invalid transition (e.g. a spike).
0	7, [31:14]	r	**Reserved;** read as 0; should be written with 0.

1) While IDLE = 1, bit field MCMP is cleared.

2) The bits in the bit fields EXPH and CURH correspond to the hall patterns at the input pins CCPOSx (x = 0, 1, 2) in the order (EXPH.2, EXPH.1, EXPH.0), (CURH.2, CURH.1, CURH.0), (CCPOS2, CCPOS.1, CCPOS0).

Register MCMCTR contains control bits for the multi-channel functionality.

MCMCTR
Multi-Channel Mode Control Register **Reset Value: 0000 0000$_H$**



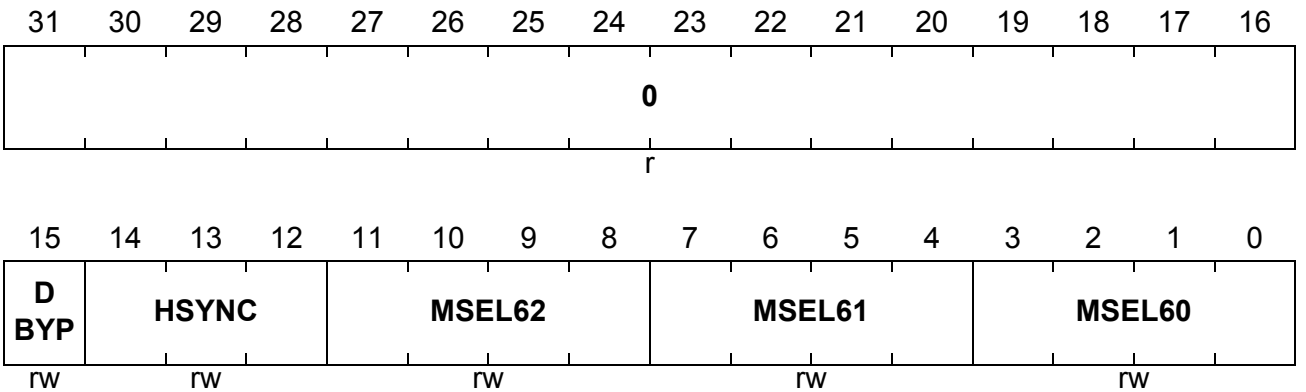
Field	Bits	Type	Description
SWSEL	[2:0]	rw	**Switching Selection** Bit field SWSEL selects one of the following trigger request sources (next multi-channel event) for the shadow transfer from MCMPS to MCMP. The trigger request is stored in the reminder flag R until the shadow transfer is done and flag R is cleared automatically with the shadow transfer. The shadow transfer takes place synchronously with an event selected in bit field SWSYN. 000 No trigger request will be generated 001 Correct hall pattern on CCPOSx detected 010 T13 period-match detected (while counting up) 011 T12 one-match (while counting down) 100 T12 channel 1 compare-match detected (phase delay function) 101 T12 period match detected (while counting up) else Reserved, no trigger request will be generated

Field	Bits	Type	Description
SWSYN	[5:4]	rw	**Switching Synchronization** Bit field SWSYN triggers the shadow transfer between MCMPS and MCMP if it has been requested before (flag R set by an event selected by SWSEL). This feature permits the synchronization of the outputs to the PWM source, that is used for modulation (T12 or T13). 00 Direct; the trigger event directly causes the shadow transfer 01 T13 zero-match triggers the shadow transfer 10 A T12 zero-match (while counting up) triggers the shadow transfer 11 Reserved; no action
0	3, [31:6]	r	**Reserved;** read as 0; should be written with 0.

Note: The generation of the shadow transfer request by hardware is only enabled if bit MCMEN = 1.

Register T12MSEL contains control bits to select the capture/compare functionality of the three channels of timer T12.

T12MSEL
T12 Capture/Compare Mode Select Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
MSEL60, MSEL61, MSEL62	[3:0], [7:4], [11:8]	rw	**Capture/Compare Mode Selection** These bit fields select the operating mode of the three timer T12 capture/compare channels. Each channel (n = 0, 1, 2) can be programmed individually either for compare or capture operation according to: 0000 Compare outputs disabled. No capture action. 0001 Compare output on pin CC6n. No capture action. 0010 Compare output on pin COUT6n. No capture action. 0011 Compare output on pins COUT6n and CC6n. 01XX Double-Register Capture modes, see **Table 29-3**. 1000 Hall Sensor mode, see **Table 29-4**. In order to enable the hall edge detection, all three MSEL6x must be programmed to Hall Sensor mode. 1001 Hysteresis-like mode, see **Table 29-4**. 101X Multi-Input Capture modes, see **Table 29-5**. 11XX Multi-Input Capture modes, see **Table 29-5**.

Field	Bits	Type	Description
HSYNC	[14:12]	rw	**Hall Synchronization** Bit field HSYNC defines the source for the sampling of the Hall input pattern and the comparison to the current and the expected Hall pattern bit fields. In all modes, a trigger by software by writing a 1 to bit SWHC is possible. 000 Any edge at one of the inputs CCPOSx (x = 0, 1, 2) triggers the sampling. 001 A T13 compare-match triggers the sampling. 010 A T13 period-match triggers the sampling. 011 The Hall sampling triggered by hardware sources is switched off. 100 A T12 period-match (while counting up) triggers the sampling. 101 A T12 one-match (while counting down) triggers the sampling. 110 A T12 compare-match of channel 0 (while counting up) triggers the sampling. 111 A T12 compare-match of channel 0 (while counting down) triggers the sampling.
DBYP	15	rw	**Delay Bypass** Bit DBYP defines if the source signal for the sampling of the Hall input pattern (selected by HSYNC) uses the dead-time counter DTC0 of timer T12 as additional delay or if the delay is bypassed. 0 The delay bypass is not active. The dead-time counter DTC0 is generating a delay after the source signal becomes active. 1 The delay bypass is active. The dead-time counter DTC0 is not used by the sampling of the Hall pattern.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

Note: In the capture modes, all edges at the CC6x inputs lead to the setting of the corresponding interrupt status flags in register IS. In order to monitor the selected capture events at the CCPOSx inputs in the multi-input capture modes, the CC6xST bits of the corresponding channel are set when detecting the selected event. The interrupt status bits and the CC6xST bits must be reset by software.

Table 29-3 Double-Register Compare Modes

Description
Double-Register Capture Modes
0100 The contents of T12 are stored in CC6nR after a rising edge and in CC6nSR after a falling edge on the input pin CC6n.
0101 The value stored in CC6nSR is copied to CC6nR after a rising edge on the input pin CC6n. The actual timer value of T12 is simultaneously stored in the shadow register CC6nSR. This feature is useful for time measurements between consecutive rising edges on pins CC6n.
0110 The value stored in CC6nSR is copied to CC6nR after a falling edge on the input pin CC6n. The actual timer value of T12 is simultaneously stored in the shadow register CC6nSR. This feature is useful for time measurements between consecutive falling edges on pins CC6n.
0111 The value stored in CC6nSR is copied to CC6nR after any edge on the input pin CC6n. The actual timer value of T12 is simultaneously stored in the shadow register CC6nSR. This feature is useful for time measurements between consecutive edges on pins CC6n.

Table 29-4 Combined T12 Modes

Description
Combined T12 Modes
1000 Hall Sensor mode: Capture mode for channel 0, compare mode for channels 1 and 2. The contents of T12 are captured into CC60 at a valid hall event (which is a reference to the actual speed). CC61 can be used for a phase delay function between hall event and output switching. CC62 can act as a time-out trigger if the expected hall event comes too late. The value ‘1000’ must be programmed to MSEL0, MSEL1, and MSEL2 if the hall signals are used. In this mode, the contents of timer T12 is captured in CC60 and T12 is reset after the detection of a valid hall event. In order to avoid noise effects, the dead-time counter channel 0 is started after an edge has been detected at the hall inputs. When reaching the value of ‘000001’, the hall inputs are sampled and the pattern comparison is done.
1001 Hysteresis-like control mode with dead time generation: The negative edge of the CCPOSx input signal is used to reset bit CC6nST. As a result, the output signals can be switched to passive state immediately and switch back to active state (with dead time) if the CCPOSx is high and the bit CC6nST is set by a compare event.

Table 29-5 Multi-Input Capture Modes

Description
Multi-Input Capture Modes
1010 The timer value of T12 is stored in CC6nR after a rising edge at the input pin CC6n. The timer value of T12 is stored in CC6nSR after a falling edge at the input pin CCPOSx.
1011 The timer value of T12 is stored in CC6nR after a falling edge at the input pin CC6n. The timer value of T12 is stored in CC6nSR after a rising edge at the input pin CCPOSx.
1100 The timer value of T12 is stored in CC6nR after a rising edge at the input pin CC6n. The timer value of T12 is stored in CC6nSR after a rising edge at the input pin CCPOSx.
1101 The timer value of T12 is stored in CC6nR after a falling edge at the input pin CC6n. The timer value of T12 is stored in CC6nSR after a falling edge at the input pin CCPOSx.
1110 The timer value of T12 is stored in CC6nR after any edge at the input pin CC6n. The timer value of T12 is stored in CC6nSR after any edge at the input pin CCPOSx.
1111 Reserved (no capture or compare action)

29.2.5 Interrupt Control Registers

Register IS contains the individual interrupt request bits. This register can only be read, write actions have no impact on the contents of this register. The software can set or reset the bits individually by writing to the register ISS (to set the bits) or to register ISR (to reset the bits).

IS
Capture/Compare Interrupt Status Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STR	IDLE	WHE	CHE	TRPS	TRPF	T13PM	T13CM	T12PM	T12OM	ICC62F	ICC62R	ICC61F	ICC61R	ICC60F	ICC60R
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
ICC60R, ICC61R, ICC62R	0, 2, 4	rh	**Capture, Compare-Match Rising Edge Flag** In compare mode, a compare-match has been detected while T12 was counting up. In capture mode, a rising edge has been detected at the input CC6x (x = 0, 1, 2). 0 The event has not yet occurred since this bit has been reset for the last time. 1 The event described above has been detected.
ICC60F, ICC61F, ICC62F	1, 3, 5	rh	**Capture, Compare-Match Falling Edge Flag** In compare mode, a compare-match has been detected while T12 was counting down. In capture mode, a falling edge has been detected at the input CC6x (x = 0, 1, 2). 0 The event has not yet occurred since this bit has been reset for the last time. 1 The event described above has been detected.

Field	Bits	Type	Description
T12OM	6	rh	**Timer T12 One-Match Flag** 0 A timer T12 one-match (while counting down) has not yet been detected since this bit has been reset for the last time. 1 A timer T12 one-match (while counting down) has been detected.
T12PM	7	rh	**Timer T12 Period-Match Flag** 0 A timer T12 period-match (while counting up) has not yet been detected since this bit has been reset for the last time. 1 A timer T12 period-match (while counting up) has been detected.
T13CM	8	rh	**Timer T13 Compare-Match Flag** 0 A timer T13 compare-match has not yet been detected since this bit has been reset for the last time. 1 A timer T13 compare-match has been detected.
T13PM	9	rh	**Timer T13 Period-Match Flag** 0 A timer T13 period-match has not yet been detected since this bit has been reset for the last time. 1 A timer T13 period-match has been detected.
TRPF	10	rh	**Trap Flag** The trap flag TRPF will be set by hardware if TRPPEN = 1 and CTRAP = 0 or by software. If TRPM2 = 0, bit TRPF is reset by hardware if the input CTRAP becomes inactive (TRPPEN = 1). If TRPM2 = 1, bit TRPF must be reset by software in order to leave the trap state. 0 The trap condition has not been detected. 1 The trap condition has been detected (input CTRAP has been 0 or by software).
TRPS[1)]	11	rh	**Trap State** 0 The trap state is not active. 1 The trap state is active. Bit TRPS is set while bit TRPF = 1. It is reset according to the mode selected in register TRPCTR.

Field	Bits	Type	Description
CHE[2]	12	rh	**Correct Hall Event** 0 A transition to a correct (= expected) hall event has not yet been detected since this bit has been reset for the last time. 1 A transition to a correct (= expected) hall event has been detected.
WHE[3]	13	rh	**Wrong Hall Event** 0 A transition to a wrong hall event (not the expected one) has not yet been detected since this bit has been reset for the last time. 1 A transition to a wrong hall event (not the expected one) has been detected.
IDLE[4]	14	rh	**IDLE State** This bit is set together with bit WHE (wrong hall event) and it must be reset by software. 0 No action 1 Bit field MCMP is cleared, the selected outputs are set to passive state
STR	15	rh	**Multi-Channel Mode Shadow Transfer Request** This bit is set when a shadow transfer from MCMOUTS to MCMOUT takes places in multi-channel mode. 0 The shadow transfer has not yet taken place 1 The shadow transfer has taken place
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) During the trap state, the selected outputs are set to the passive state. The logic level driven during the passive state is defined by the corresponding bit in register PSLR. Bit TRPS = 1 and TRPF = 0 can occur if the trap condition is no longer active but the selected synchronization has not yet taken place.

2) On every valid hall edge, the contents of EXPH are compared with the pattern on pin CCPOSx and if equal bit CHE is set.

3) On every valid hall edge, the contents of EXPH are compared with the pattern on pin CCPOSx. If both compares (CURH and EXPH with CCPOSx) are not true, bit WHE (wrong hall event) is set.

4) Bit field MCMP is hold to 0 by hardware as long as IDLE = 1.

Note: Not all bits in register IS can generate an interrupt. Other status bits have been added, which have a similar structure for their set and reset actions.

Note: The interrupt generation is independent from the value of the bits in register IS, e.g. the interrupt will be generated (if enabled) even if the corresponding bit is already set. The trigger for an interrupt generation is the detection of a set condition (by hardware or software) for the corresponding bit in register IS.

Note: In compare mode (and hall mode), the timer-related interrupts are only generated while the timer is running (TxR = 1). In capture mode, the capture interrupts are also generated while the timer T12 is stopped.

Register ISS contains the individual interrupt request set bits required to generate a CCU6 interrupt request by software.

ISS

Capture/Compare Interrupt Status Set Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S STR	**S IDLE**	**S WHE**	**S CHE**	**S WHC**	**S TRPF**	**S T13PM**	**S T13CM**	**S T12PM**	**S T12OM**	**S CC62F**	**S CC62R**	**S CC61F**	**S CC61R**	**S CC60F**	**S CC60R**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
SCC60R	0	w	**Set Capture, Compare-Match Rising Edge Flag** 0 No action 1 Bit ICC60R in register IS will be set
SCC60F	1	w	**Set Capture, Compare-Match Falling Edge Flag** 0 No action 1 Bit ICC60F in register IS will be set
SCC61R	2	w	**Set Capture, Compare-Match Rising Edge Flag** 0 No action 1 Bit ICC61R in register IS will be set
SCC61F	3	w	**Set Capture, Compare-Match Falling Edge Flag** 0 No action 1 Bit ICC61F in register IS will be set
SCC62R	4	w	**Set Capture, Compare-Match Rising Edge Flag** 0 No action 1 Bit ICC62R in register IS will be set
SCC62F	5	w	**Set Capture, Compare-Match Falling Edge Flag** 0 No action 1 Bit ICC62F in register IS will be set

Field	Bits	Type	Description
ST12OM	6	w	**Set Timer T12 One-Match Flag** 0 No action 1 Bit T12OM in register IS will be set
ST12PM	7	w	**Set Timer T12 Period-Match Flag** 0 No action 1 Bit T12PM in register IS will be set
ST13CM	8	w	**Set Timer T13 Compare-Match Flag** 0 No action 1 Bit T13CM in register IS will be set
ST13PM	9	w	**Set Timer T13 Period-Match Flag** 0 No action 1 Bit T13PM in register IS will be set
STRPF	10	w	**Set Trap Flag** 0 No action 1 Bits TRPF and TRPS in register IS will be set
SWHC	11	w	**Software Hall Compare** 0 No action 1 The Hall compare action is triggered
SCHE	12	w	**Set Correct Hall Event Flag** 0 No action 1 Bit CHE in register IS will be set
SWHE	13	w	**Set Wrong Hall Event Flag** 0 No action 1 Bit WHE in register IS will be set
SIDLE	14	w	**Set IDLE Flag** 0 No action 1 Bit IDLE in register IS will be set
SSTR	15	w	**Set STR Flag** 0 No action 1 Bit STR in register IS will be set
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

Note: If the setting by hardware of the corresponding flags can lead to an interrupt, the setting by software has the same effect.

Register ISR contains the individual interrupt request reset the corresponding flags by software.

ISR

Capture/Compare Interrupt Status Reset Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R STR	R IDLE	R WHE	R CHE	0	R TRPF	R T13PM	R T13CM	R T12PM	R T12OM	R CC62F	R CC62R	R CC61F	R CC61R	R CC60F	R CC60R
w	w	w	w	r	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RCC60R	0	w	**Reset Capture, Compare-Match Rising Edge Flag** 0 No action 1 Bit ICC60R in register IS will be reset
RCC60F	1	w	**Reset Capture, Compare-Match Falling Edge Flag** 0 No action 1 Bit ICC60F in register IS will be reset
RCC61R	2	w	**Reset Capture, Compare-Match Rising Edge Flag** 0 No action 1 Bit ICC61R in register IS will be reset
RCC61F	3	w	**Reset Capture, Compare-Match Falling Edge Flag** 0 No action 1 Bit ICC61F in register IS will be reset
RCC62R	4	w	**Reset Capture, Compare-Match Rising Edge Flag** 0 No action 1 Bit ICC62R in register IS will be reset
RCC62F	5	w	**Reset Capture, Compare-Match Falling Edge Flag** 0 No action 1 Bit ICC62F in register IS will be reset
RT12OM	6	w	**Reset Timer T12 One-Match Flag** 0 No action 1 Bit T12OM in register IS will be reset

Field	Bits	Type	Description
RT12PM	7	w	**Reset Timer T12 Period-Match Flag** 0 No action 1 Bit T12PM in register IS will be reset
RT13CM	8	w	**Reset Timer T13 Compare-Match Flag** 0 No action 1 Bit T13CM in register IS will be reset
RT13PM	9	w	**Reset Timer T13 Period-Match Flag** 0 No action 1 Bit T13PM in register IS will be reset
RTRPF	10	w	**Reset Trap Flag** 0 No action 1 Bit TRPF in register IS will be reset (not taken into account while input $\overline{\text{CTRAP}}$ = 0 and TRPPEN = 1)
RCHE	12	w	**Reset Correct Hall Event Flag** 0 No action 1 Bit CHE in register IS will be reset
RWHE	13	w	**Reset Wrong Hall Event Flag** 0 No action 1 Bit WHE in register IS will be reset
RIDLE	14	w	**Reset IDLE Flag** 0 No action 1 Bit IDLE in register IS will be reset
RSTR	15	w	**Reset STR Flag** 0 No action 1 Bit STR in register IS will be reset
0	11, [31:16]	r	**Reserved;** read as 0; should be written with 0.

Register IEN contains the interrupt enable bits and a control bit to enable the automatic idle function in the case of a wrong hall pattern.

IEN
Capture/Compare Interrupt Enable Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENSTR	ENIDLE	ENWHE	ENCHE	0	ENTRPF	ENT13PM	ENT13CM	ENT12PM	ENT12OM	ENCC62F	ENCC62R	ENCC61F	ENCC61R	ENCC60F	ENCC60R
rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENCC60R	0	rw	**Capture, Compare-Match Rising Edge Interrupt Enable for Channel 0** 0 No interrupt will be generated if the set condition for bit ICC60R in register IS occurs. 1 An interrupt will be generated if the set condition for bit ICC60R in register IS occurs. The interrupt line, which will be activated, is selected by bit field INPCC60.
ENCC60F	1	rw	**Capture, Compare-Match Falling Edge Interrupt Enable for Channel 0** 0 No interrupt will be generated if the set condition for bit ICC60F in register IS occurs. 1 An interrupt will be generated if the set condition for bit ICC60F in register IS occurs. The interrupt line, which will be activated, is selected by bit field INPCC60.
ENCC61R	2	rw	**Capture, Compare-Match Rising Edge Interrupt Enable for Channel 1** 0 No interrupt will be generated if the set condition for bit ICC61R in register IS occurs. 1 An interrupt will be generated if the set condition for bit ICC61R in register IS occurs. The interrupt line, which will be activated, is selected by bit field INPCC61.

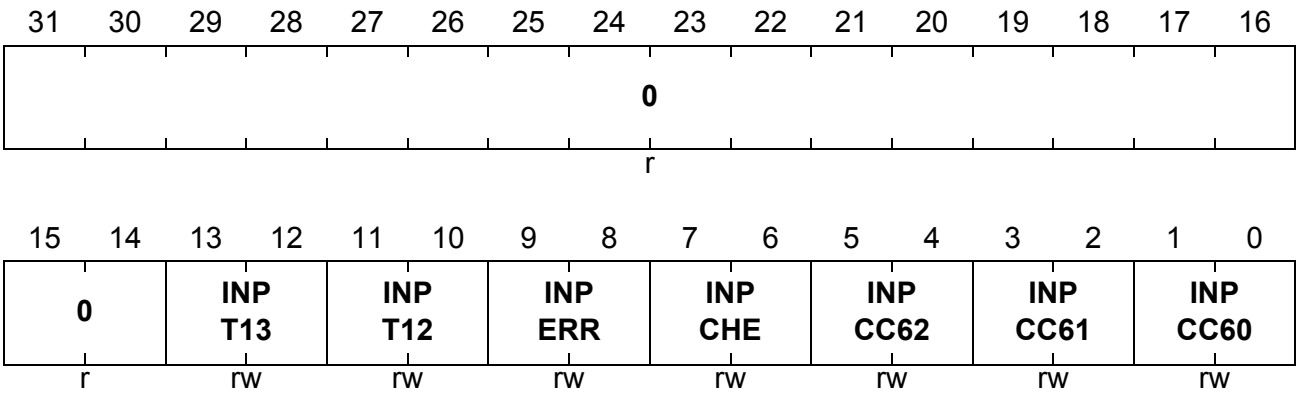
Field	Bits	Type	Description
ENCC61F	3	rw	**Capture, Compare-Match Falling Edge Interrupt Enable for Channel 1** 0 No interrupt will be generated if the set condition for bit ICC61F in register IS occurs. 1 An interrupt will be generated if the set condition for bit ICC61F in register IS occurs. The interrupt line, which will be activated, is selected by bit field INPCC61.
ENCC62R	4	rw	**Capture, Compare-Match Rising Edge Interrupt Enable for Channel 2** 0 No interrupt will be generated if the set condition for bit ICC62R in register IS occurs. 1 An interrupt will be generated if the set condition for bit ICC62R in register IS occurs. The interrupt line, which will be activated, is selected by bit field INPCC62.
ENCC62F	5	rw	**Capture, Compare-Match Falling Edge Interrupt Enable for Channel 2** 0 No interrupt will be generated if the set condition for bit ICC62F in register IS occurs. 1 An interrupt will be generated if the set condition for bit ICC62F in register IS occurs. The interrupt line, which will be activated, is selected by bit field INPCC62.
ENT12OM	6	rw	**Enable Interrupt for T12 One-Match** 0 No interrupt will be generated if the set condition for bit T12OM in register IS occurs. 1 An interrupt will be generated if the set condition for bit T12OM in register IS occurs. The interrupt line, which will be activated, is selected by bit field INPT12.
ENT12PM	7	rw	**Enable Interrupt for T12 Period-Match** 0 No interrupt will be generated if the set condition for bit T12PM in register IS occurs. 1 An interrupt will be generated if the set condition for bit T12PM in register IS occurs. The interrupt line, which will be activated, is selected by bit field INPT12.

Field	Bits	Type	Description
ENT13CM	8	rw	**Enable Interrupt for T13 Compare-Match** 0 No interrupt will be generated if the set condition for bit T13CM in register IS occurs. 1 An interrupt will be generated if the set condition for bit T13CM in register IS occurs. The interrupt line, which will be activated, is selected by bit field INPT13.
ENT13PM	9	rw	**Enable Interrupt for T13 Period-Match** 0 No interrupt will be generated if the set condition for bit T13PM in register IS occurs. 1 An interrupt will be generated if the set condition for bit T13PM in register IS occurs. The interrupt line, which will be activated, is selected by bit field INPT13.
ENTRPF	10	rw	**Enable Interrupt for Trap Flag** 0 No interrupt will be generated if the set condition for bit TRPF in register IS occurs. 1 An interrupt will be generated if the set condition for bit TRPF in register IS occurs. The interrupt line, which will be activated, is selected by bit field INPERR.
ENCHE	12	rw	**Enable Interrupt for Correct Hall Event** 0 No interrupt will be generated if the set condition for bit CHE in register IS occurs. 1 An interrupt will be generated if the set condition for bit CHE in register IS occurs. The interrupt line, which will be activated, is selected by bit field INPCHE.
ENWHE	13	rw	**Enable Interrupt for Wrong Hall Event** 0 No interrupt will be generated if the set condition for bit WHE in register IS occurs. 1 An interrupt will be generated if the set condition for bit WHE in register IS occurs. The interrupt line, which will be activated, is selected by bit field INPERR.

Field	Bits	Type	Description
ENIDLE	14	rw	**Enable Idle** This bit enables the automatic entering of the idle state (bit IDLE will be set) after a wrong hall event has been detected (bit WHE is set). During the idle state, the bit field MCMP is automatically cleared. 0 The bit IDLE is not automatically set when a wrong hall event is detected. 1 The bit IDLE is automatically set when a wrong hall event is detected.
ENSTR	15	rw	**Enable Multi-Channel Mode Shadow Transfer Interrupt** 0 No interrupt will be generated if the set condition for bit STR in register IS occurs. 1 An interrupt will be generated if the set condition for bit STR in register IS occurs. The interrupt line, which will be activated, is selected by bit field INPCHE.
0	11, [31:16]	r	**Reserved;** read as 0; should be written with 0.

Register INP contains the interrupt node pointer bits, thus allowing for flexible interrupt handling.

INP
Capture/Compare Interrupt Node Pointer Register **Reset Value: 0000 3940$_H$**



Field	Bits	Type	Description
INPCC60	[1:0]	rw	**Interrupt Node Pointer for Channel 0 Interrupts** This bit field defines the interrupt output line, which is activated due to a set condition for bit ICC60R (if enabled by bit ENCC60R) or for bit ICC60F (if enabled by bit ENCC60F). 00 Interrupt output line SR0 is selected 01 Interrupt output line SR1 is selected 10 Interrupt output line SR2 is selected 11 Interrupt output line SR3 is selected
INPCC61	[3:2]	rw	**Interrupt Node Pointer for Channel 1 Interrupts** This bit field defines the interrupt output line, which is activated due to a set condition for bit ICC61R (if enabled by bit ENCC61R) or for bit ICC61F (if enabled by bit ENCC61F). 00 Interrupt output line SR0 is selected 01 Interrupt output line SR1 is selected 10 Interrupt output line SR2 is selected 11 Interrupt output line SR3 is selected

Field	Bits	Type	Description
INPCC62	[5:4]	rw	**Interrupt Node Pointer for Channel 2 Interrupts** This bit field defines the interrupt output line, which is activated due to a set condition for bit ICC62R (if enabled by bit ENCC62R) or for bit ICC62F (if enabled by bit ENCC62F). 00 Interrupt output line SR0 is selected 01 Interrupt output line SR1 is selected 10 Interrupt output line SR2 is selected 11 Interrupt output line SR3 is selected
INPCHE	[7:6]	rw	**Interrupt Node Pointer for the CHE Interrupt** This bit field defines the interrupt output line, which is activated due to a set condition for bit CHE (if enabled by bit ENCHE) of for bit STR (if enabled by bit ENSTR). 00 Interrupt output line SR0 is selected 01 Interrupt output line SR1 is selected 10 Interrupt output line SR2 is selected 11 Interrupt output line SR3 is selected
INPERR	[9:8]	rw	**Interrupt Node Pointer for Error Interrupts** This bit field defines the interrupt output line, which is activated due to a set condition for bit TRPF (if enabled by bit ENTRPF) or for bit WHE (if enabled by bit ENWHE). 00 Interrupt output line SR0 is selected 01 Interrupt output line SR1 is selected 10 Interrupt output line SR2 is selected 11 Interrupt output line SR3 is selected
INPT12	[11:10]	rw	**Interrupt Node Pointer for Timer12 Interrupts** This bit field defines the interrupt output line, which is activated due to a set condition for bit T12OM (if enabled by bit ENT12OM) or for bit T12PM (if enabled by bit ENT12PM). 00 Interrupt output line SR0 is selected 01 Interrupt output line SR1 is selected 10 Interrupt output line SR2 is selected 11 Interrupt output line SR3 is selected

Field	Bits	Type	Description
INPT13	[13:12]	rw	**Interrupt Node Pointer for Timer13 Interrupt** This bit field defines the interrupt output line, which is activated due to a set condition for bit T13CM (if enabled by bit ENT13CM) or for bit T13PM (if enabled by bit ENT13PM). 00 Interrupt output line SR0 is selected 01 Interrupt output line SR1 is selected 10 Interrupt output line SR2 is selected 11 Interrupt output line SR3 is selected
0	[31:14]	r	**Reserved;** read as 0; should be written with 0.

29.3 CCU60/CCU61 Module Implementation

This section describes CCU60/CCU61 module interfaces with the clock control, port connections, interrupt control, and address decoding.

29.3.1 Interfaces of the CCU6 Modules

Figure 29-32 shows the TC1130 specific implementation details and interconnections of the CCU60/CCU61 modules. Both of the CCU6 modules are further supplied by clock control, interrupt control, address decoding, and port control logic. One DMA request can be generated by each CCU6 module.

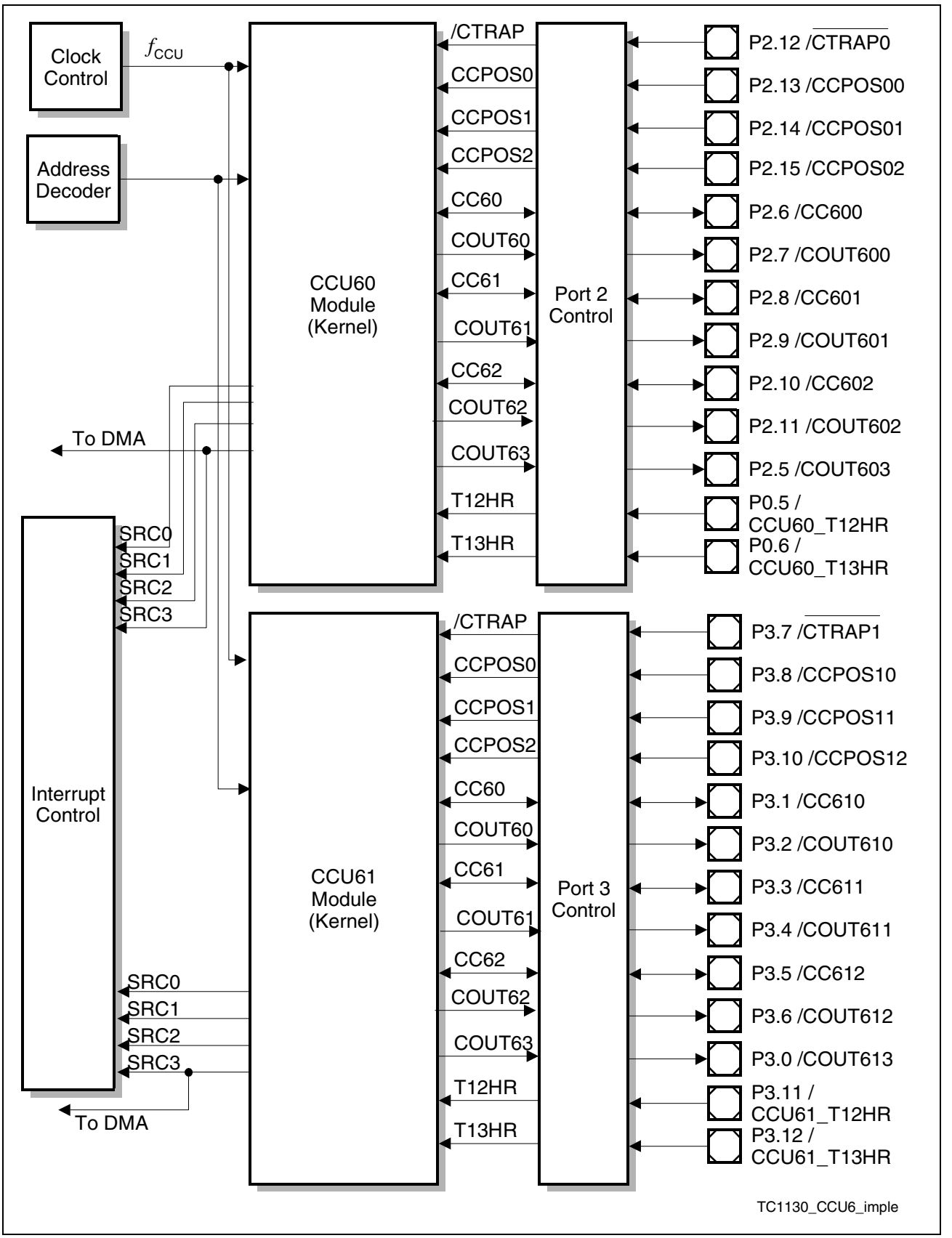


Figure 29-32 CCU6 Module Implementation and Interconnections

29.3.2 CCU60/CCU61 Module Related External Registers

Figure 29-33 summarizes the module related external registers which are required for CCU60/CCU61 programming (see also **Figure 29-31** for the module kernel specific registers).

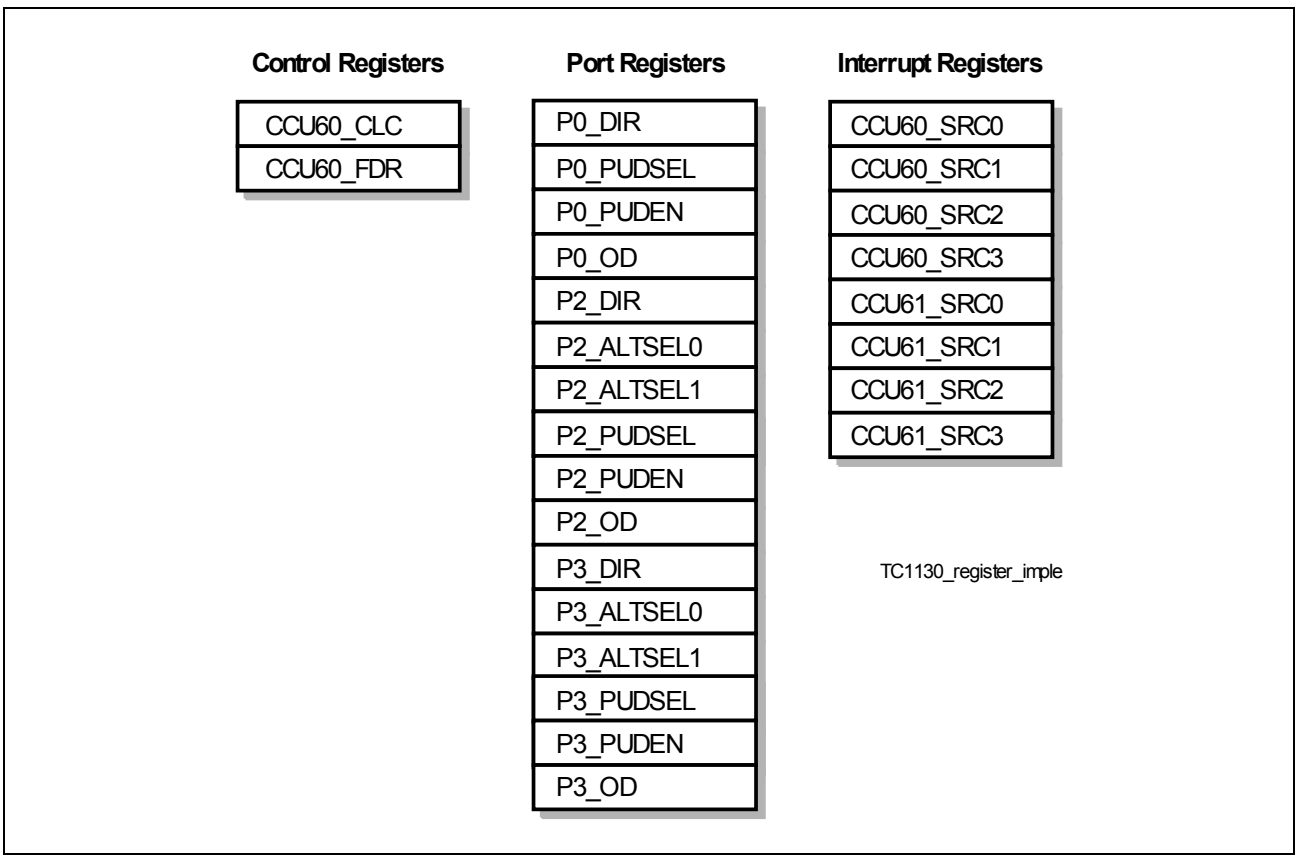


Figure 29-33 CCU60/CCU61 Implementation Specific Special Function Registers

29.3.2.1 Clock Control

The CCU6 modules are provided each with two clock signals:

- f_{CLC}:
  This is the module clock that is used inside the CCU6 kernel for control purposes such as e.g. for clocking of control logic and register operations. The frequency of f_{CLC} is always identical to the system clock frequency f_{SYS}. The clock control register CCU60_CLC allows the f_{CLC} to be enabled/disabled under certain conditions.
- f_{CCU}:
  This is the module clock that is used in the CCU6 modules as input of the prescalers for T12 and T13. The fractional divider register CCU60_FDR controls the frequency of f_{CCU} and allows it to be enabled/disabled independently of f_{CLC}.

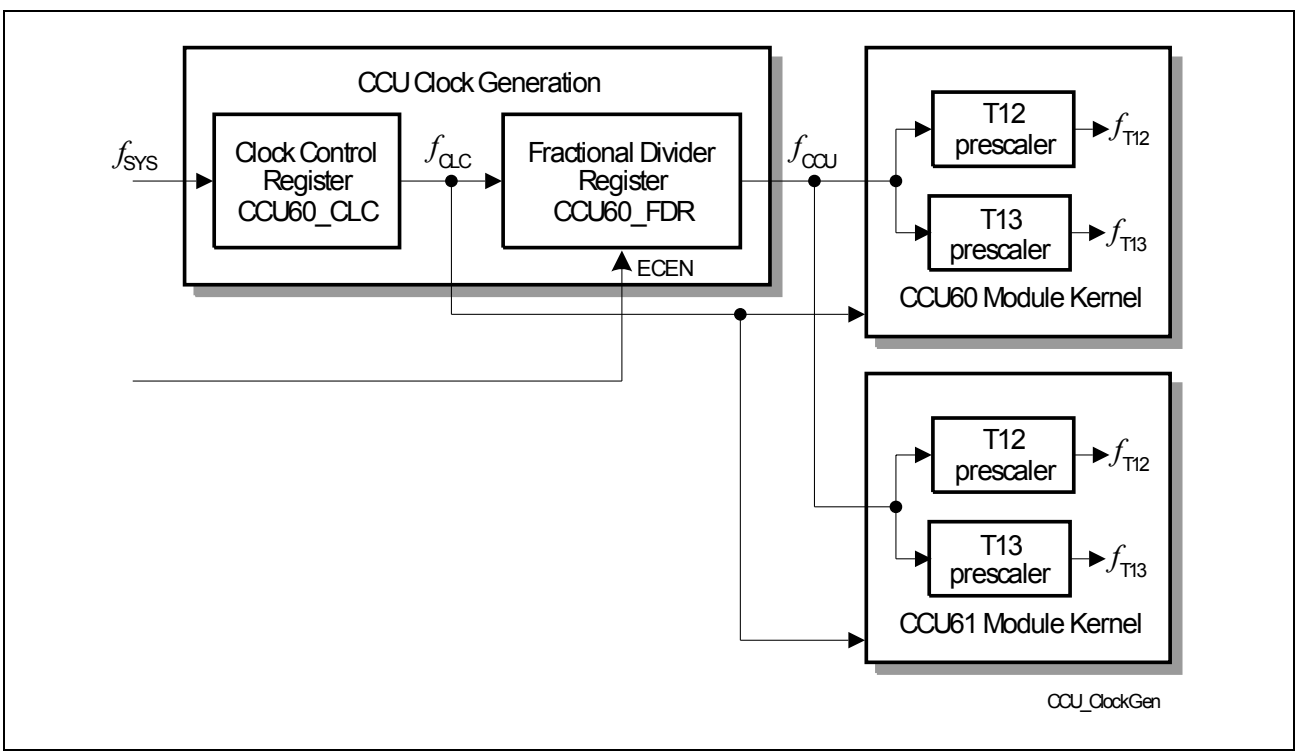


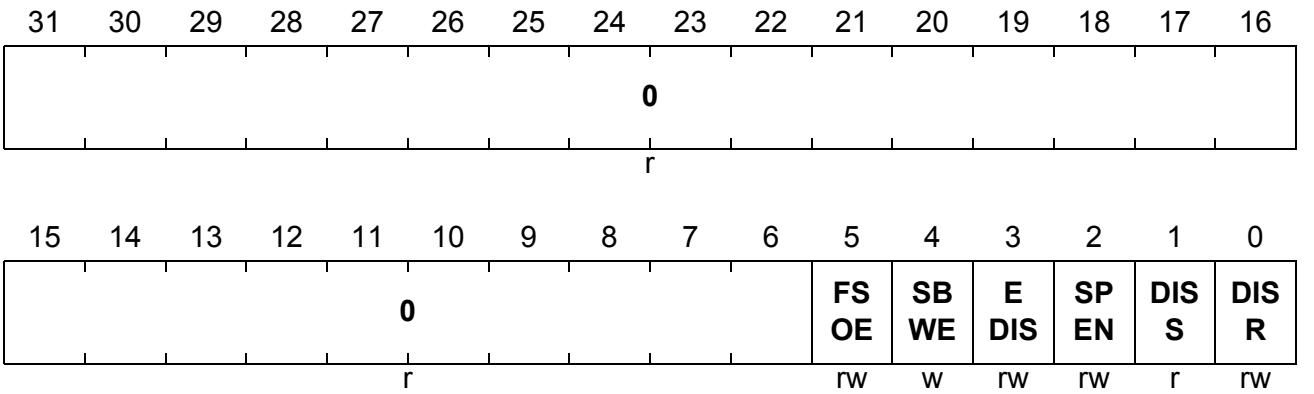
Figure 29-34 CCU6 Clock Generation

The synchronization input line ECEN of the fractional divider is connected to output signal CAN_INT_O15 of the MultiCAN module.

29.3.2.2 Clock Control Register

The common clock control register allows the programmer to control (enable/disable) the clock signals to the CCU6 modules under certain conditions.

CCU60_CLC
CCU60 Clock Control Register **Reset Value: 0000 0003$_H$**

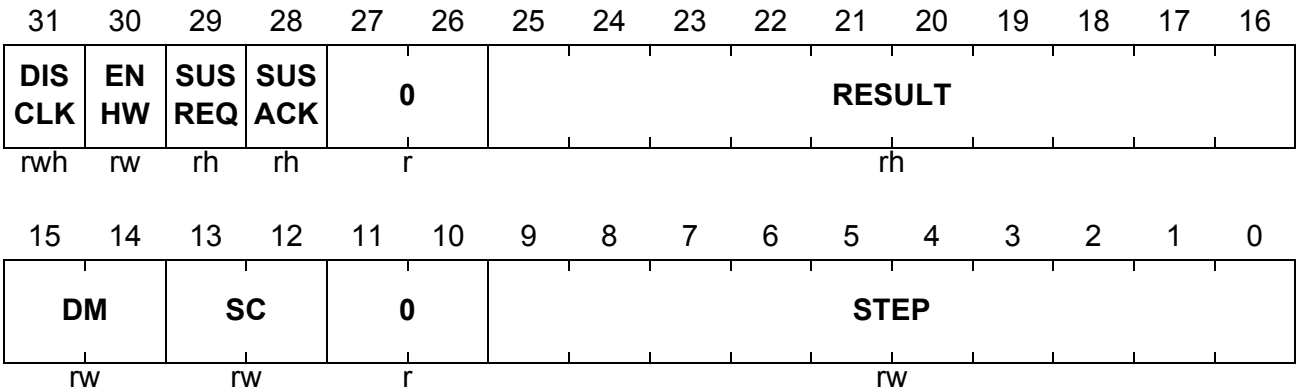


Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the module.
SPEN	2	rw	**Module Suspend Enable for OCDS** Used for enabling the suspend mode.
EDIS	3	rw	**External Request Disable** Used for controlling the external clock disable request.
SBWE	4	w	**Module Suspend Bit Write Enable for OCDS** Defines whether SPEN and FSOE are write protected.
FSOE	5	rw	**Fast Switch Off Enable** Used for fast clock switch off in OCDS suspend mode.
0	[31:6]	r	**Reserved;** read as 0; should be written with 0.

29.3.2.3 Fractional Divider Register

The common fractional divider register controls the clock rate of the shift clock f_{CCU}.

CCU60_FDR
CCU60 Fractional Divider Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
STEP	[9:0]	rw	**Step Value** Reload or addition value for RESULT.
SC	[13:12]	rw	**Suspend Control** This bit field defines the behavior of the fractional divider in suspend mode.
DM	[15:14]	rw	**Divider Mode** This bit field selects normal divider mode or fractional divider mode.
RESULT	[25:16]	rh	**Result Value** Bit fields for the addition result.
SUSACK	28	rh	**Suspend Mode Acknowledge** Indicates state of SPNDACK signal.
SUSREQ	29	rh	**Suspend Mode Request** Indicates state of SPND signal.
ENHW	30	rw	**Enable Hardware Clock Control** Controls operation of ECEN input and DISCLK bit.
DISCLK	31	rwh	**Disable Clock** Hardware controlled disable for f_{OUT} signal.
0	[11:10], [27:26]	rw	**Reserved;** read as 0; should be written with 0.

29.3.3 Port Control

The interconnections between the CCU6 modules and the I/O lines/pins are controlled by software in the port logics. The CCU60/CCU61 I/O functionality must be selected by the following port control operations (additionally to the PISEL programming):

- Input/output function selection (DIR registers)
- Alternate function selection (ALTSEL0 and ALTSEL1 registers)
- Input/Output driver characteristic control (PUDSEL, PUDEN and OD registers)

The CCU60/CCU61 port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as pull-up/down devices, port direction (input/output), open-drain, and alternate output selections. The I/O lines for the CCU60/CCU61 module are controlled by the port input/output control registers of Port 0, Port 2 and Port 3.

Table 29-6 shows how bits and bit fields must be programmed for the required I/O functionality of the CCU6 I/O lines.

Table 29-6 CCU60 and CCU61 I/O Line Selection and Setup

Module	Port Lines	Input/Output Control Register Bits	I/O
CCU60	P2.5/COUT603	P2_DIR.P5 = 1_B	Output
		P2_ALTSEL0.P5 = 0_B	
		P2_ALTSEL1.P5 = 1_B	
	P2.6/CC600	P2_DIR.P6 = 0_B	Input
		P2_DIR.P6 = 1_B	Output
		P2_ALTSEL0.P6 = 0_B	
		P2_ALTSEL1.P6 = 1_B	
	P2.7/COUT600	P2_DIR.P7 = 1_B	Output
		P2_ALTSEL0.P7 = 0_B	
		P2_ALTSEL1.P7 = 1_B	
	P2.8/CC601	P2_DIR.P8 = 0_B	Input
		P2_DIR.P8 = 1_B	Output
		P2_ALTSEL0.P8 = 0_B	
		P2_ALTSEL1.P8 = 1_B	
	P2.9/COUT601	P2_DIR.P9 = 1_B	Output
		P2_ALTSEL0.P9 = 0_B	
		P2_ALTSEL1.P9 = 1_B	
	P2.10/CC602	P2_DIR.P10 = 0_B	Input
		P2_DIR.P10 = 1_B	Output
		P2_ALTSEL0.P10 = 0_B	
		P2_ALTSEL1.P10 = 1_B	
	P2.11/COUT602	P2_DIR.P11 = 1_B	Output
		P2_ALTSEL0.P11 = 0_B	
		P2_ALTSEL1.P11 = 1_B	
	P2.12/CTRAP0	P2_DIR.P12 = 0_B	Input
	P2.13/CCPOS00	P2_DIR.P13 = 0_B	Input
	P2.14/CCPOS01	P2_DIR.P14 = 0_B	Input
	P2.15/CCPOS02	P2_DIR.P15 = 0_B	Input
	P0.5/CC60_T12HR	P0_DIR.P5 = 0_B	Input
	P0.6/CC60_T13HR	P0_DIR.P6 = 0_B	Input

Table 29-6 CCU60 and CCU61 I/O Line Selection and Setup (cont'd)

Module	Port Lines	Input/Output Control Register Bits	I/O
CCU61	P3.0/COUT613	P3_DIR.P0 = 1_B	Output
		P3_ALTSEL0.P0 = 1_B	
		P3_ALTSEL1.P0 = 0_B	
	P3.1/CC610	P3_DIR.P1 = 0_B	Input
		P3_DIR.P1 = 1_B	Output
		P3_ALTSEL0.P1 = 1_B	
		P3_ALTSEL1.P1 = 0_B	
	P3.2/COUT610	P3_DIR.P2 = 1_B	Output
		P3_ALTSEL0.P2 = 1_B	
		P3_ALTSEL1.P2 = 0_B	
	P3.3/CC611	P3_DIR.P3 = 0_B	Input
		P3_DIR.P3 = 1_B	Output
		P3_ALTSEL0.P3 = 1_B	
		P3_ALTSEL1.P3 = 0_B	
	P3.4/COUT611	P3_DIR.P4 = 1_B	Output
		P3_ALTSEL0.P4 = 1_B	
		P3_ALTSEL1.P4 = 0_B	
	P3.5/CC612	P3_DIR.P5 = 0_B	Input
		P3_DIR.P5 = 1_B	Output
		P3_ALTSEL0.P5 = 1_B	
		P3_ALTSEL1.P5 = 0_B	
	P3.6/COUT612	P3_DIR.P6 = 1_B	Output
		P3_ALTSEL0.P6 = 1_B	
		P3_ALTSEL1.P6 = 0_B	
	P3.7/CTRAP1	P3_DIR.P7 = 0_B	Input
	P3.8/CCPOS10	P3_DIR.P8 = 0_B	Input
	P3.9/CCPOS11	P3_DIR.P9 = 0_B	Input
	P3.10/CCPOS12	P3_DIR.P10 = 0_B	Input
	P3.11/CC61_T12HR	P3_DIR.P11 = 0_B	Input
	P3.12/CC62_T13HR	P3_DIR.P12 = 0_B	Input

P0_DIR
Port 0 Direction Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 5, 6)**	n	rw	**Port 0 Pin 5, 6 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CCU60 I/O port control.

P2_DIR
Port 2 Direction Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 5-15)**	n	rw	**Port 2 Pin 5-15 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CCU60 I/O port control.

P3_DIR
Port 3 Direction Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-12)**	n	rw	**Port 3 Pin 0-12 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CCU61 I/O port control.

P2_ALTSELn (n = 0, 1)
Port 2 Alternate Select Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 29-7 Function of the Bits P2_ALTSEL0.Pn and P2_ALTSEL1.Pn (n = 5-11)[1)]

P2_ALTSEL0.Pn	P2_ALTSEL1.Pn	Function
0	1	Alternate Select 0

1) Shaded bits and bit field are don't care for CCU60 I/O port control.

P3_ALTSELn (n = 0, 1)
Port 3 Alternate Select Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 29-8 Function of the Bits P3_ALTSEL0.Pn and P3_ALTSEL1.Pn (n = 0-6)[1)]

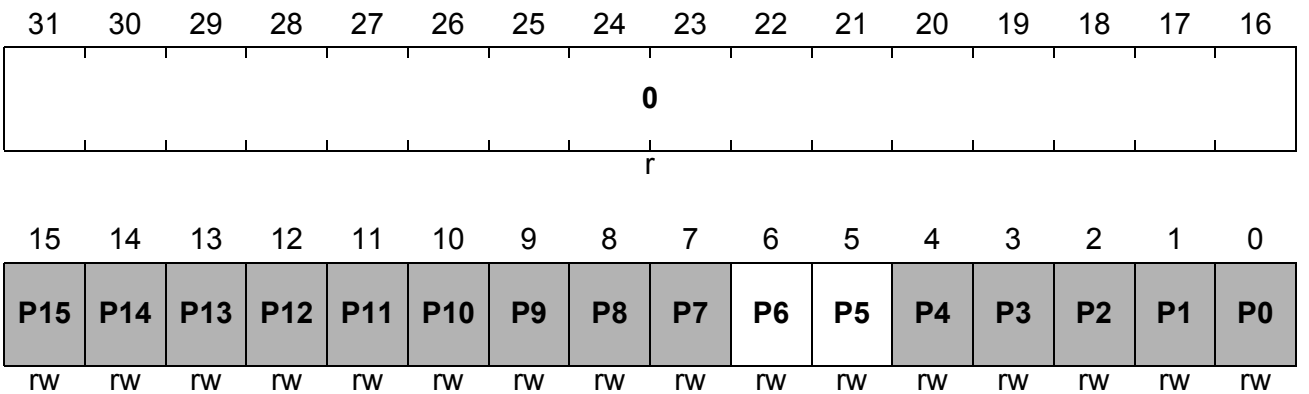
P3_ALTSEL0.Pn	P3_ALTSEL1.Pn	Function
1	0	Alternate Select 1

1) Shaded bits and bit field are don't care for CCU61 I/O port control.

The CCU60/CCU61 ports also offer the possibility to configure the following output characteristics:

- Push/pull (optional pull-up/pull-down)
- Open drain with internal pull-up
- Open drain with external pull-up

P0_PUDSEL
Port 0 Pull-Up/Pull-Down Select Register **Reset Value: 0000 FFFF$_H$**



Field	Bits	Type	Description
Pn **(n = 5, 6)**	n	rw	**Pull-Up/Pull-Down Select Port 0 Bit n**[1)] 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CCU60 I/O port control.

P2_PUDSEL
Port 2 Pull-Up/Pull-Down Select Register

Reset Value: 0000 0FFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
r				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 5-11)**	n	rw	**Pull-Up/Pull-Down Select Port 2 Bit n**[1)] 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:12]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CCU60 I/O port control.

P3_PUDSEL
Port 3 Pull-Up/Pull-Down Select Register

Reset Value: 0000 FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-12)**	n	rw	**Pull-Up/Pull-Down Select Port 3 Bit n**[1)] 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CCU61 I/O port control.

P0_PUDEN
Port 0 Pull-Up/Pull-Down Enable Register

Reset Value: 0000 FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 5, 6)**	n	rw	**Pull-Up/Pull-Down Enable at Port 0 Bit n**[1)] 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CCU60 I/O port control.

P2_PUDEN
Port 2 Pull-Up/Pull-Down Enable Register

Reset Value: 0000 0FFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
r				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 5-11)**	n	rw	**Pull-Up/Pull-Down Enable at Port 2 Bit n**[1)] 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:12]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CCU60 I/O port control.

P3_PUDEN
Port 3 Pull-Up/Pull-Down Enable Register

Reset Value: 0000 FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-12)**	n	rw	**Pull-Up/Pull-Down Enable at Port 3 Bit n**[1)] 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CCU61 I/O port control.

P0_OD
Port 0 Open Drain Control Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 5, 6)**	n	rw	**Port 2 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CCU60 I/O port control.

P2_OD
Port 2 Open Drain Control Register **Reset Value: 0000 F000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	**P14**	**P13**	**P12**	**P11**	**P10**	**P9**	**P8**	**P7**	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 5-11)**	n	rw	**Port 2 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CCU60 I/O port control.

P3_OD
Port 3 Open Drain Control Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-6)**	n	rw	**Port 3 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CCU61 I/O port control.

29.3.3.1 Service Request Registers

Each CCU6 module has four service request outputs. The interrupt output lines SRC[3:0] of each CCU6 module are connected to service request nodes. The service request control registers are described as below.

The interrupt output lines SRC[3] of each CCU6 module are also connected to the DMA module. The request assignment is described in the DMA implementation chapter.

CCU60_SRCx (x = 0-3)
CCU60 Service Request Control Register x **Reset Value: 0000 0000$_H$**
CCU61_SRCx (x = 0-3)
CCU61 Service Request Control Register x **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETR	CLRR	SRR	SRE	0	TOS	0		SRPN							
w	w	rh	rw	r	rw	r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number**
TOS	10	rw	**Type of Service Control**
SRE	12	rw	**Service Request Enable**
SRR	13	rh	**Service Request Flag**
CLRR	14	w	**Request Clear Bit**
SETR	15	w	**Request Set Bit**
0	[9:8], 11, [31:16]	r	**Reserved;** read as 0; should be written with 0.

Note: Further details on interrupt handling and processing are described in chapter "Interrupt System" of the TC1130 System Units User's Manual. The detailed DMA request connections are defined in chapter "Direct Memory Access Controller" of the TC1130 System Units User's Manual.

29.3.4 DMA Requests

The DMA request lines of the CCU60/CCU61 modules become active whenever its related interrupt line is activated. The DMA request lines are connected to the DMA controller as shown in **Table 29-9**.

Table 29-9 DMA Request Lines of CCU60/CCU61

Module	Related SSC Interrupt	DMA Request Line	Description
CCU60	SRC3	CCU60_SRC3	CCU60 DMA Request
CCU61	SRC3	CCU61_SRC3	CCU61 DMA Request

29.3.5 CCU60/CCU61 Register Address Ranges

In the TC1130, the registers of the CCU6 modules are located in the following address ranges:

- CCU60 module: Module Base Address = F000 2000_H
  Module End Address = F000 $20FF_H$
- CCU61 module: Module Base Address = F000 2100_H
  Module End Address = F000 $21FF_H$
- Absolute Register Address = Module Base Address + Offset Address
  (offset addresses see **Table 29-2**)

Note: The complete and detailed address map of the CCU60/CCU61 modules is described in the chapter “Register Overview” of the TC1130 System Units User’s Manual.

30 Controller Area Network (MultiCAN) Controller

This chapter describes the MultiCAN controller of the TC1130. It contains the following sections:

- Functional description of the MultiCAN Kernel (see **Section 30.1**)
- Register descriptions of all MultiCAN Kernel specific registers (see **Section 30.2**)
- TC1130 implementation specific details and registers of the MultiCAN controller (port connections and control, interrupt control, address decoding, clock control, see **Section 30.3**)

Note: The MultiCAN kernel register names described in this chapter will be referenced in the TC1130 User's Manual by the module name prefix "CAN_".

30.1 MultiCAN Kernel Description

30.1.1 Overview

The MultiCAN module contains 4 Full-CAN nodes operating independently or exchanging data and remote frames via a gateway function. Transmission and reception of CAN frames is handled in accordance to CAN specification V2.0part B (active). Each CAN node can receive and transmit standard frames with 11-bit identifiers as well as extended frames with 29-bit identifiers.

All CAN nodes share a common set of message objects, where each message object may be individually allocated to one of the CAN nodes. Besides serving as a storage container for incoming and outgoing frames, message objects may be combined to build gateways between the CAN nodes or to setup a FIFO buffer.

The message objects are organized in double chained lists, where each CAN node has its own list of message objects. A CAN node stores frames only into message objects that are allocated to the list of the CAN node. It only transmits messages from objects of this list.

A powerful, command driven list controller performs all list operations.

The bit timings for the CAN nodes are derived from the peripheral clock (f_{CAN}) and are programmable up to a data rate of 1 MBaud. A pair of receive and transmit pins connects each CAN node to a bus transceiver.

Features

- Compliant to ISO 11898
- CAN functionality according to CAN specification V2.0 B active
- Dedicated control registers are provided for each CAN node
- A data transfer rate up to 1 MBaud is supported
- Flexible and powerful message transfer control and error handling capabilities are implemented
- Advanced CAN bus bit timing analysis and baud rate detection can be performed for each CAN node via the frame counter
- Full-CAN functionality: A set of 128 message objects can be individually
  - allocated (assigned) to any CAN node
  - configured as transmit or receive object
  - setup to handle frames with 11-bit or 29-bit identifier
  - counted or assigned a timestamp via a frame counter
  - configured to remote monitoring mode
- Advanced Acceptance Filtering:
  - Each message object provides an individual acceptance mask to filter incoming frames

  - A message object can be configured to accept only standard or only extended frames or to accept both standard and extended frames
  - Message objects can be grouped into 4 priority classes
  - The selection of the message to be transmitted first can be performed on the basis of frame identifier, IDE bit and RTR bit according to CAN arbitration rules
- Advanced Message Object Functionality:
  - Message Objects can be combined to build FIFO message buffers of arbitrary size, which is only limited by the total number of message objects
  - Message objects can be linked to form a gateway to automatically transfer frames between 2 different CAN buses. A single gateway can link any two CAN nodes. An arbitrary number of gateways may be defined
- Advanced Data Management:
  - The Message objects are organized in double chained lists
  - List reorganizations may be performed any time, even during full operation of the CAN nodes
  - A powerful, command driven list controller manages the organization of the list structure and ensures consistency of the list
  - Message FIFOs are based on the list structure and can easily be scaled in size during CAN operation
  - Static Allocation Commands offer compatibility with TwinCAN applications, which are not list based
- Advanced Interrupt Handling:
  - Up to 16 interrupt output lines are available. Most interrupt requests can be individually routed to one of the 16 interrupt output lines
  - Message postprocessing notifications can be flexibly aggregated into a dedicated register field of 256 notification bits

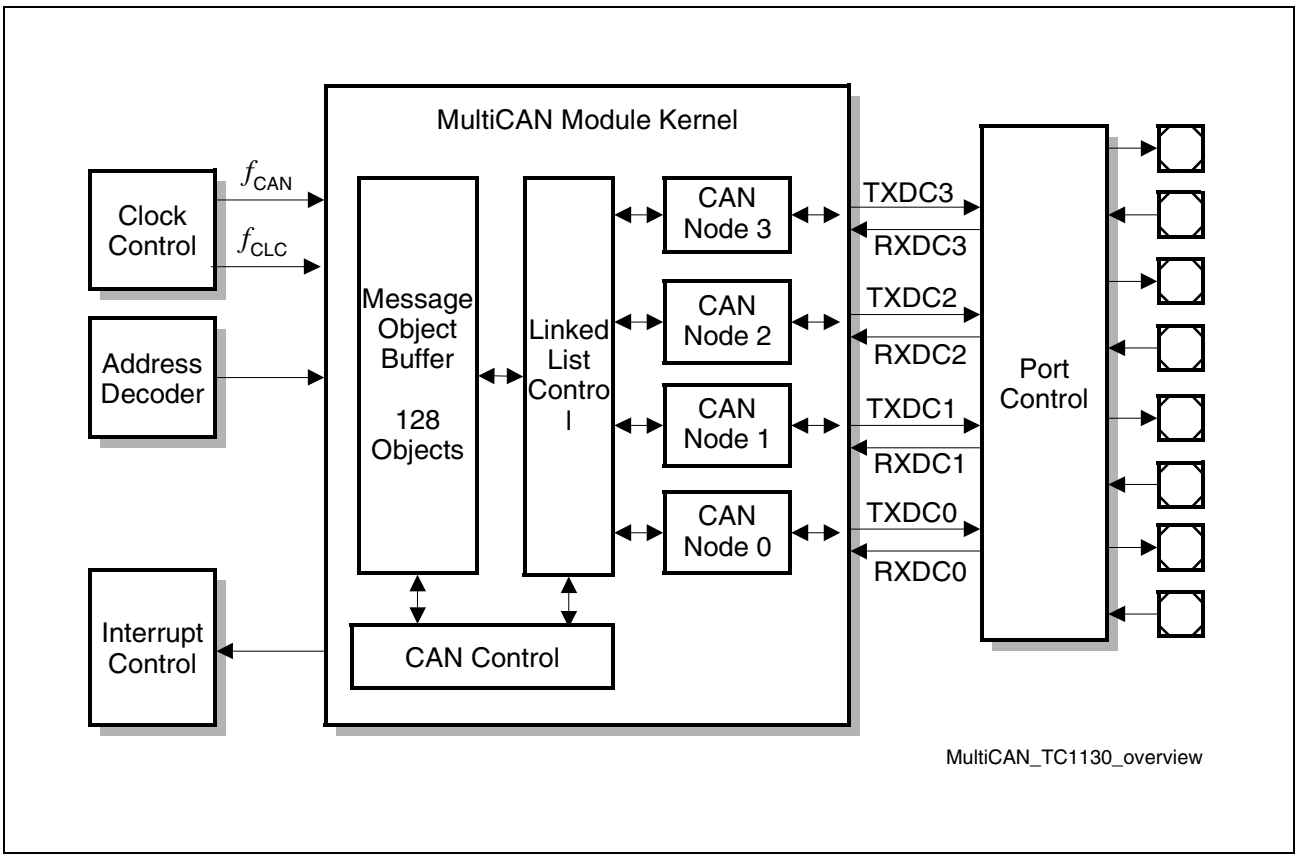


Figure 30-1 Overview of the MultiCAN

Table 30-1 defines constants that are used throughout the MultiCAN specification. These are fixed values for a given MultiCAN implementation.

Table 30-1 Fixed Module Constants

Constant	Value	Description
n_objects	**128**	**Number of Message Objects** n_objects denotes the total amount of message objects available.
n_interrupts	**16**	**Number of Interrupt Nodes** n_interrupts denotes the total number of interrupt outputs available.
n_pendings	**256**	**Number of Message Pending Bits** n_pendings denotes the number of message pending bits available. The number of message pending registers is given by n_pendings/32.

Table 30-1 Fixed Module Constants (cont'd)

Constant	Value	Description
n_lists	**8**	**Number of Lists** n_lists denotes the total number of lists available for allocation of message number.
n_nodes	**4**	**Number of CAN Nodes Available** n_nodes denotes the total number of CAN nodes available. As each CAN node has its own list in addition to the list of un-allocated elements, the relation n_nodes < n_lists is true.

30.1.2 Module Structure

Figure 30-2 shows the general structure of the MultiCAN module with 4 CAN nodes.

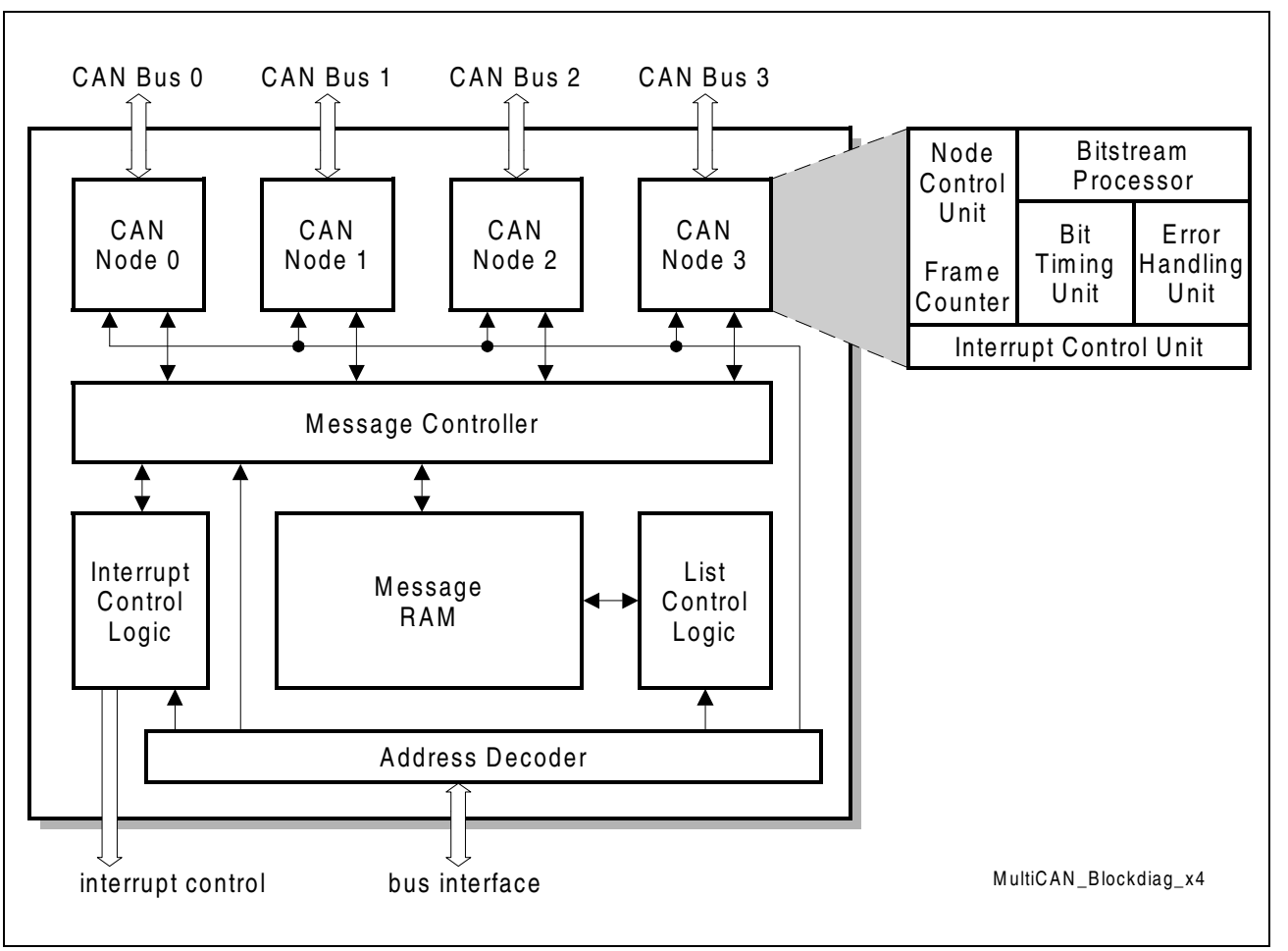


Figure 30-2 MultiCAN Block Diagram with 4 CAN Nodes

Each CAN node consists of several subunits as described in **Table 30-2**:

Table 30-2 Subunits of CAN Nodes

Subunit	Description
Bit Stream Processor	The Bit Stream Processor performs data, remote, error and overload frame processing according to the ISO 11898 standard. This includes conversion between the serial data stream and the input/output shift registers.
Bit Timing Unit	The Bit Timing Unit defines the length of a bit time and the location of the sample point according to the user settings, taking into account propagation delays and phase shift errors. The Bit Timing Unit also performs resynchronization.
Error Handling Unit	The Error Handling Unit manages the receive and transmit error counter. According to the contents of both counters the CAN node is set into an “Error Active”, “Error Passive” or “Bus-Off” state.
Node Control Unit	The Node Control Unit coordinates the operation of the CAN node: • Enable/disable CAN transfer of the node • Enable/disable and generate node specific events that lead to an interrupt request (CAN bus errors, successful frame transfers etc.) • Administration of the Frame Counter

Message Controller

The message controller handles the exchange of CAN frames between the CAN nodes and the message objects which are stored in the Message RAM. It performs:

- Receive Acceptance filtering to determine the correct message object for storing of a received CAN frame
- Transmit Acceptance Filtering to determine the message object to be transmitted first, individually for each CAN node
- Content transfer between message objects and the CAN nodes, taking into account the status/control bits of the message objects
- Handling of the FIFO buffering and Gateway functionality
- Aggregation of message pending notification bits

List Controller

The list controller performs all operations that lead to a modification of the double chained message object lists. Only the list controller is allowed to modify the list structure. The allocation/deallocation or reallocation of a message object can be

requested via a user command interface (command panel). The list controller state machine then performs the requested command autonomously.

30.1.3 CAN Node Control

Each CAN node may be configured and run independently from the other CAN nodes. To this end each CAN node is equipped with an individual set of SFR registers to control and to monitor the CAN node.

30.1.3.1 Bit Timing

According to ISO 11898 standard, a CAN bit time is subdivided into different segments (**Figure 30-3**). Each segment consists of multiples of a time quantum t_q. The magnitude of t_q is adjusted by bit NBTR.BRP and by bit NBTR.DIV8, both controlling the baud rate prescaler. The baud rate prescaler is driven by the MultiCAN module clock f_{CAN}.

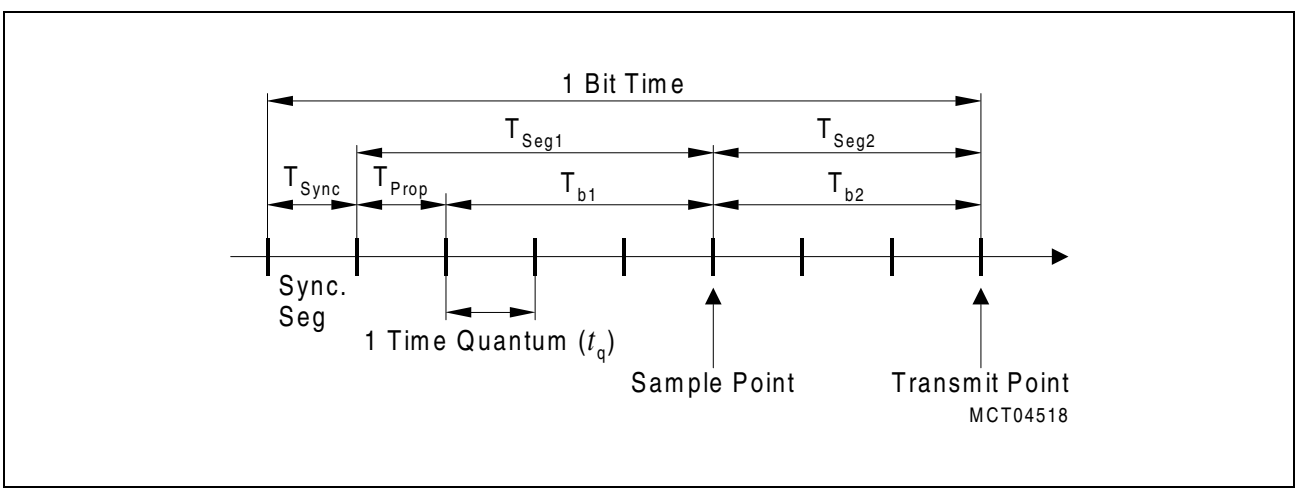


Figure 30-3 CAN Bus Bit Timing Standard

The Synchronization Segment (T_{Sync}) allows a phase synchronization between transmitter and receiver time base. The Synchronization Segment length is always 1 t_q. The Propagation Time Segment (T_{Prop}) takes into account the physical propagation delay in the transmitter output driver, on the CAN bus line and in the transceiver circuit. For a working collision detect mechanism, T_{Prop} has to be two times the sum of all propagation delay quantities rounded up to a multiple of t_q. The Phase Buffer Segments 1 and 2 (T_{b1}, T_{b2}) before and after the signal sample point are used to compensate a mismatch between transmitter and receiver clock phase detected in the synchronization segment.

The maximum number of time quanta allowed for resynchronization is defined by bit NBTR.SJW. The Propagation Time Segment and the Phase Buffer Segment 1 are combined to parameter TSeg1, which is defined by bit NBTR.TSEG1. A minimum of 3 time quanta is requested by the ISO standard. Parameter TSeg2, which is defined by bit NBTR.TSEG2, covers the Phase Buffer Segment 2. A minimum of 2 time quanta is

requested by the ISO standard. According ISO standard, a CAN bit time, calculated as the sum of T_{Sync}, T_{Seg1} and T_{Seg2}, must not fall below 8 time quanta.

Calculation of the bit time:

t_q = (BRP + 1) / f_{CAN}, if DIV8 = 0

= 8 × (BRP + 1) / f_{CAN}, if DIV8 = 1

T_{Sync} = 1 t_q

T_{Seg1} = (TSEG1 + 1) × t_q (min. 3 t_q)

T_{Seg2} = (TSEG2 + 1) × t_q (min. 2 t_q)

bit time = T_{Sync} + T_{Seg1} + T_{Seg2} (min. 8 t_q)

To compensate phase shifts between clocks of different CAN controllers, the CAN controller has to synchronize on any edge from the recessive to the dominant bus level. If the hard synchronization is enabled (at the start of frame), the bit time is restarted at the synchronization segment. Otherwise, the resynchronization jump width T_{SJW} defines the maximum number of time quanta a bit time may be shortened or lengthened by one resynchronization. The value of SJW is programmed in the CAN Node Bit Timing Register NBTR.

T_{SJW} = (SJW + 1) × t_q

$T_{Seg1} \geq T_{SJW} + T_{prop}$

$T_{Seg2} \geq T_{SJW}$

The maximum relative tolerance for f_{CAN} depends on the Phase Buffer Segments and the resynchronization jump width.

$df_{CAN} \leq \min(T_{b1}, T_{b2}) / 2 \times (13 \times \text{bit time} - T_{b2})$ AND

$df_{CAN} \leq T_{SJW} / 20 \times \text{bit time}$

A valid CAN bit timing must be written to the CAN Node Bit Timing Register NBTR before resetting the bit NCR.INIT, i.e. before enabling the operation of the CAN node.

The Node Bit Timing Register NBTR may be written only if bit NCR.CCE (Configuration Change Enable) is set.

30.1.3.2 CAN Error Handling

The Error Handling Unit of the CAN node is responsible for the fault confinement of the CAN device. Its two counters, the Receive Error Counter and the Transmit Error Counter (control register NECNT), are incremented and decremented by commands from the Bit Stream Processor. If the Bit Stream Processor itself detects an error while a transmit operation is running, the Transmit Error Counter is incremented by 8. An increment of 1 is used, when the error condition was reported by an external CAN node via an error frame generation. For error analysis, the transfer direction of the disturbed message and the node, recognizing the transfer error, are indicated in the control register NECNT of the respective CAN node. According to the values of the error counters, the CAN node is set into the states “error active”, “error passive” and “bus-off”.

The CAN node is in error active state, if both error counters are below the error passive limit of 128. It is in error passive state, if at least one of the error counters equals or exceeds 128.

The bus-off state is activated if the Transmit Error Counter equals or exceeds the bus-off limit of 256. This state is reported by flag BOFF in the NSR status register of the CAN node. The device remains in this state, until the bus-off recovery sequence is finished. Additionally, there is the bit EWRN in the NSR status register, which is set, if at least one of the error counters equals or exceeds the error warning limit defined by bit field EWRNLVL in the control registers NECNT of the CAN node. Bit NSR.EWRN is reset if both error counters fall below the error warning limit again.

30.1.3.3 CAN Frame Counter

Each CAN node is equipped with a frame counter. This enables the counting of transmitted/received CAN frames. It also helps obtain information about the exact time instant of starting the transmission or reception of a frame by the CAN node. CAN frame counting/bit time counting is performed by a 16-bit counter which is controlled by register NFCR of the respective CAN node. Bit field CFSEL of register NFCR defines the operation mode of the frame counter:

- Frame Count Mode: The frame counter is incremented after the successful transmission and/or reception of a CAN frame. The incremented value is stored to the CFC field of register NFCR and copied to the CFCVAL field of the Interrupt Pointer Register MOIPR of the message object involved in the transfer.
- Time Stamp Mode: The frame counter is incremented with the beginning of a new bit time. When the transmission/reception of a frame starts, the value of the frame counter is captured and stored to the CFC field of register NFCR. After the successful transfer of the frame, the captured value is copied to the CFCVAL field of the Interrupt Pointer Register MOIPR of the message object involved in the transfer.
- Bit Timing Mode: Used for baud rate detection and analysis of the bit timing (**Section 30.1.5.3**).

30.1.3.4 CAN Node Interrupts

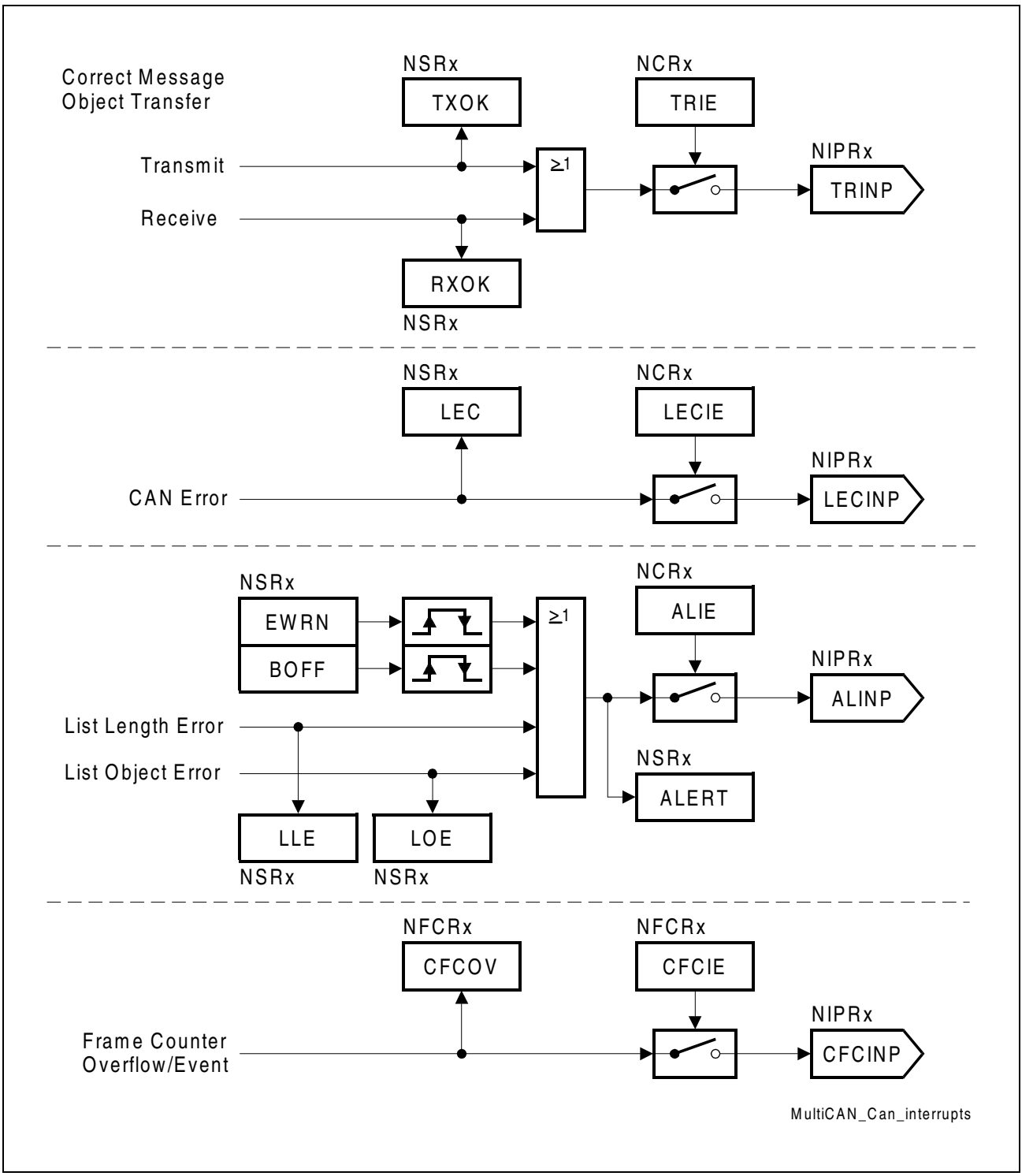


Figure 30-4 CAN Node Interrupts

Each CAN node is equipped with four interrupt sources to generate an interrupt request upon:

- The successful transmission/reception of a frame,

- An overflow of the frame counter (frame count mode/time stamp mode) or a bit timing measurement event (bit timing mode),
- An error related to the CAN node.

30.1.4 Message Object List Structure

30.1.4.1 Basics

The message objects of the MultiCAN module are organized in double chained lists, where each message object has a pointer to the previous message object in the list as well as a pointer to the next message object in the list. The MultiCAN module provides 16 different lists, where each object is allocated to one of these lists. A 4-bit LIST bit field in the Message Object Control Register MOCTR indicates the list to which the respective message object is currently allocated. In the example of **Figure 30-5** three message objects are allocated to the list with list index 2.

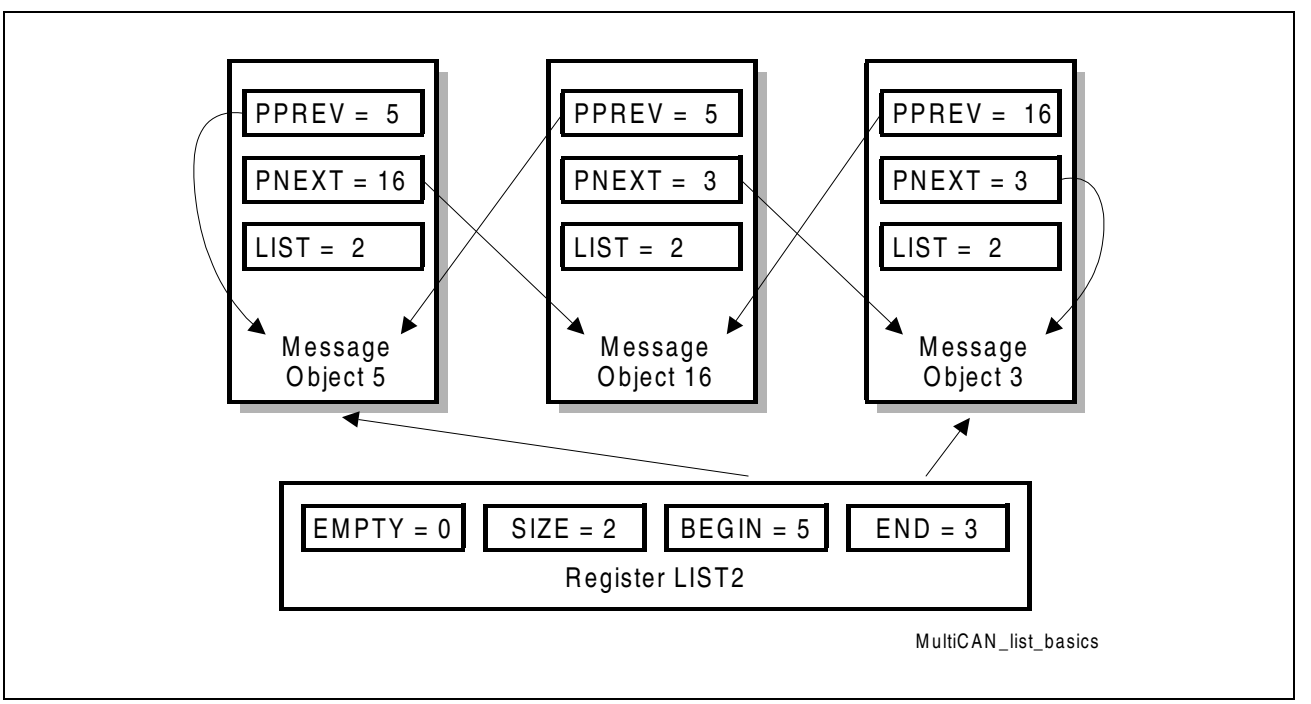


Figure 30-5 Example Allocation of Message Objects to a List

The BEGIN field of the List Register LIST points to the first element in the list (object 5 in the example) whereas the LIST.END field points to the last element in the list (object 3 in the example). The number of elements in the list is indicated in the LIST.SIZE field (#elements = SIZE + 1, thus SIZE = 2 for the 3 elements of the example). The EMPTY bit indicates a list with no elements (EMPTY = 0 in the example, as the list is not empty).

Each message object has a pointer PNEXT (located in the Message Object Control Register) that points to the next message object in the list and a pointer PREV that points to the previous message object in the list. PPREV of the first message object points to

the object itself because the first object has no predecessor (in the example object 5 is the first object, indicated by PPREV = 5). PNEXT of the last message object also points to the object itself because the last element has no successor (in the example object 3 is the last object, indicated by PNEXT = 3).

Each message object also has a 4-bit LIST field (located in the Message Object Control Register) which shows list index of the list to which the object is currently allocated (the objects of the example are allocated to list 2, thus LIST = 2).

30.1.4.2 List of Unallocated Elements

The list with list index 0 has a special meaning: It is the list of all unallocated elements. An element is called unallocated if and only if it belongs to list zero. It is called allocated if and only if it belongs to one of the other lists.

After reset all message objects are unallocated, i.e. belong to the list of unallocated elements. The initial allocation of the message objects within the list of unallocated objects is ordered by message number, i.e. the predecessor of message object n is object n-1 and the successor of object n is object n+1.

30.1.4.3 Connection to the CAN Nodes

Each CAN node is linked to exactly one unique list of message objects. A CAN node performs message transfer only with the message objects that are allocated to the list of the CAN node. This is illustrated in **Figure 30-6** which shows the MultiCAN module with 4 CAN nodes. Frames that are received on a CAN node may only be stored in one of the message objects that belongs to the CAN node and frames to be transmitted on a CAN node are selected only from the message objects that are allocated to that node, as indicated by the vertical arrows.

In any configuration of the MultiCAN module there are more lists than CAN nodes, which means that some lists are not linked to a CAN node. A Message object that is allocated to one of this list may not receive messages directly from a CAN node and it may not transmit messages. **Table 30-7** defines the usage of all lists.

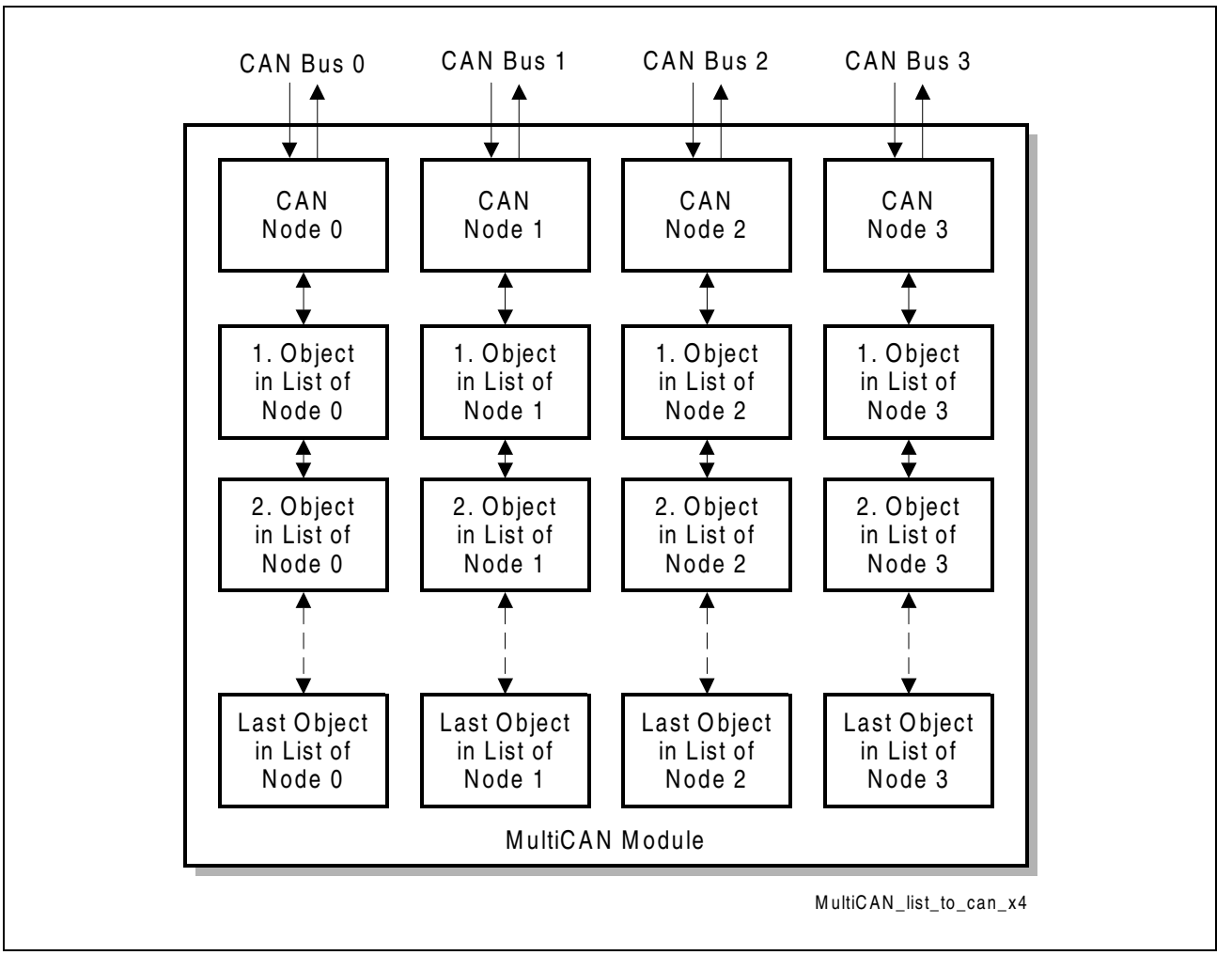

CAN Bus 0
CAN Bus 1
CAN Bus 2
CAN Bus 3
CAN Node 0
CAN Node 1
CAN Node 2
CAN Node 3
1. Object in List of Node 0
1. Object in List of Node 1
1. Object in List of Node 2
1. Object in List of Node 3
2. Object in List of Node 0
2. Object in List of Node 1
2. Object in List of Node 2
2. Object in List of Node 3
Last Object in List of Node 0
Last Object in List of Node 1
Last Object in List of Node 2
Last Object in List of Node 3
MultiCAN Module
MultiCAN_list_to_can_x4

Figure 30-6 Message Objects Linked to CAN Nodes

30.1.4.4 List Command Panel

The list structure may not be modified directly by means of write accesses to the LIST registers and the PPREV, PNEXT and LIST fields in the message objects as they are read-only. The management of the list structure is performed by and limited to the list controller unit inside the MultiCAN module. The list controller is controlled via a command panel which allows the user to issue list allocation commands to the list controller. The list controller basically serves two purposes:

1. Ensure that all operations that modify the list structure result in a consistent list structure.
2. Present maximum comfort and flexibility to the user.

The list controller and the associated command panel allows the programmer to concentrate on the final properties of the list, which are characterized by the allocation of message objects to a CAN node and the ordering relation between objects which are allocated to the same list. The process of list (re-)building is left to the list controller.

A panel command is started by writing the respective command code (see **Table 30-6**) into to the PANCMD field of the panel control register PANCTR. The corresponding command arguments must be written to PANCTR.PANAR1 and PANCTR.PANAR2 before writing the command code or latest together with the command code in a single 32-bit write access to the panel control register (only possible within 32-bit system environments).

With the write of a valid command code the PANCTR.BUSY flag becomes active (BUSY = 1) and the control panel registers are locked, which means that write accesses to the Panel Control Register are ignored. The PANCTR.BUSY flag remains active and the control panel remains locked until the execution of the requested command is completed.

When the issued command is a dynamic allocation which takes an element from the list of unallocated objects, then also the PANCTR.RBUSY bit becomes active together with the PANCTR. BUSY bit (PANCTR.RBUSY = PANCTR.BUSY = 1) to indicate that PANCTR.PANAR1 and PANCTR.PANAR2 are going to be updated by the list controller:

1. The message number of the message object taken from the list of unallocated elements is written to PANCTR.PANAR1.
2. An error status is posted to bit 7 of PANAR2 (Bit 7 = ERR). If ERR = 1 then the list of unallocated elements was empty and the command is aborted. If ERR = 0 then the list was not empty and the command will be performed successfully.

The results are written before the list controller starts the actual allocation process. As soon as the results are available, PANCTR.RBUSY becomes inactive (PANCTR.RBUSY = 0) again, while PANCTR.BUSY still remains active until completion of the command. This allows the user to setup the new message object while it is still in the process of list allocation. The access to message objects is not limited during ongoing list operations. However, any access to a register resource located inside the RAM delays the ongoing allocation process by one access cycle.

As soon as the command is done the BUSY flag becomes inactive (BUSY = 0) and write accesses to the Panel Control Register are enabled again. Also the NOP command code is automatically written to the CMD field of the Panel Control Register. A new command may be started any time during BUSY inactive.

All fields of the Panel Control Register except BUSY and RBUSY may be written by the user. This allows the Panel Control Register to be saved and restored if the Command Panel is used within independent (mutually interruptible) interrupt routines. If this is the case, then any task that uses the Command Panel and that may interrupt another task also using the Command Panel should poll the BUSY flag until is becomes inactive and save the whole PANCTR register to a save memory location before issuing a command. At the end it should restore PANCTR from the said memory location.

Before a message object that is allocated to the list of an active CAN node is moved to another list or to another position within the same list, bit MSGVAL (“Message Valid”) should be cleared in the Message Object Control Register of the message object.

30.1.5 CAN Node Analysis Features

30.1.5.1 Analyze Mode

CAN Analyze Mode allows the CAN traffic to be monitored without affecting the logical state of the CAN bus.

CAN Analyze Mode is selected by setting bit CALM in the Node Control Register. CAN Analyze Mode may be selected for each CAN node individually.

In CAN Analyze Mode the transmit pin of the CAN node is held on recessive level. The CAN node may receive frames (data-, remote-, and error frames) but is not allowed to transmit. Active error frames are sent recessive. Received data/remote frames are not acknowledged (i.e. acknowledge slot is sent recessive), but will be received and stored in matching message objects as long as there is any other node that acknowledges the frame.

All message object functionality is available, but no transmit request will be executed.

30.1.5.2 Loop-back Mode

The MultiCAN module provides a loop-back mode to enable an in-system test of the MultiCAN module as well as the development of CAN driver software without access to an external CAN bus.

The loop-back feature consists of an internal CAN bus (inside the MultiCAN module) and a bus select switch for each CAN node (**Figure 30-7**). With the switch each CAN node can be wired either to the internal CAN bus (loop-back mode activated) or the external CAN bus, i.e. the transmit- and receive pins (normal operation). The CAN bus which is currently not selected is driven recessive, i.e. the transmit pin is held at 1 and the receive pin is ignored by the CAN nodes which are in loop-back mode.

Loop-back Mode is selected individually for each CAN node by setting bit LBM in the respective Node Port Control Register. All CAN nodes that are in loop-back mode may communicate on the internal CAN bus without affecting the normal operation of the other CAN nodes which are not in loop-back mode.

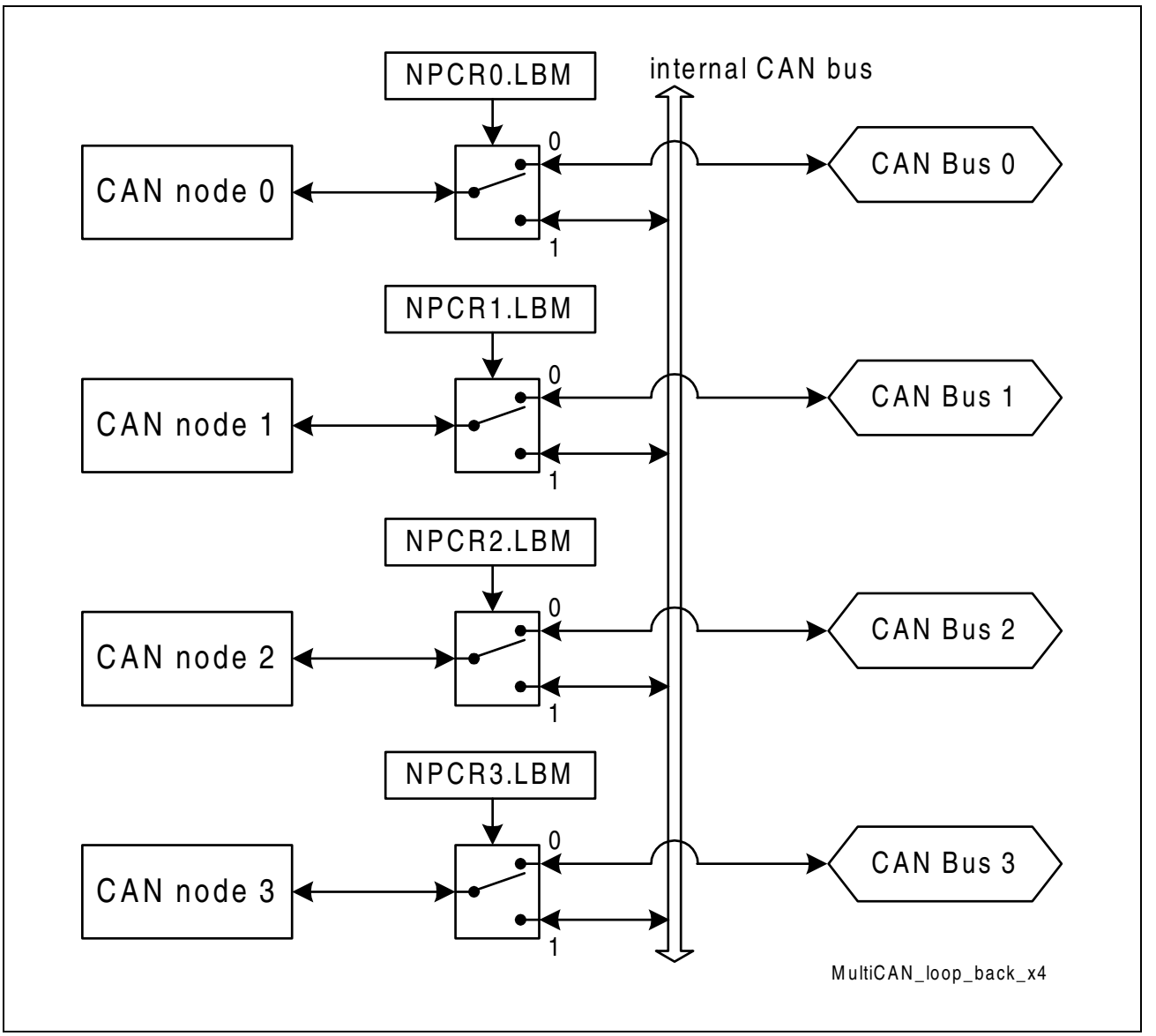


Figure 30-7 Loop-back Mode for 4 CAN Nodes

30.1.5.3 Bit Timing Analysis

For each CAN node detailed analysis of the bit timing can be performed by means of using dedicated analysis modes of the CAN frame counter. The bit timing analysis functionality of the frame counter may be used for automatic detection of the CAN baud rate as well as for the analysis of the timing of the CAN network.

Bit timing analysis for a CAN node is selected by CFMOD = 10_B (Bit Timing Mode) in the CAN Node Frame Counter Register NFCR.

Bit timing analysis does not affect the operation of the CAN node.

The measurement results are written to the NFCR.CFC field. Whenever NFCR.CFC is updated in Bit Timing Mode, then also the NFCR.CFCOV bit is set in order to indicate the update event. If NFCR.CFCIE is set then also an interrupt request is generated, where four CAN nodes i = 0 to 3 the interrupt request is generated on interrupt node i.

Automatic Baud Rate Detection

Automatic baud rate detection requires to measure the time between the observation of subsequent dominant edges on the CAN bus. This measurement is automatically performed if NFCR.CFSEL = 000_B in the CAN Node Frame Counter Register. With each dominant edge monitored on the CAN receive input the time (measured in clock cycles) between this edge and the most recent dominant edge is stored in the NFCR.CFC field.

Synchronization Analysis

The bit time synchronization is monitored if NFCR.CFSEL = 010_B. The time between the first dominant edge and the sample point is measured and stored in NFCR.CFC. The bit timing synchronization offset may be derived from this time as the first edge after the sample point triggers synchronization and there is only one synchronization between consecutive sample points.

Synchronization Analysis may be used to fine tune the baud rate during reception of the first CAN frame with the measured baud rate.

Driver Delay Measurement

The delay between a transmitted edge and the corresponding received edge is measured with NFCR.CFSEL = 011_B (dominant to dominant) and NFCR.CFSEL = 100_B (recessive to recessive). These delays indicate the time needed to represent a new bit value on the physical implementation of the CAN bus.

30.1.6 Message Acceptance Filtering

30.1.6.1 Receive Acceptance Filtering

When a message object is received on a CAN node, then a unique message object is determined in which the received frame will be stored upon successful frame reception. A message object qualifies for the reception of a frame if and only if the following conditions are fulfilled:

1. The message object is allocated to the list of the CAN node on which the frame is received.
2. MSGVAL is set in the Message Object Control Register.
3. RXEN is set in the Message Object Control Register.
4. The MOCTR.DIR bit in the Message Object Control Register equals the RTR bit of the received frame. If MOCTR.DIR = 1 (transmit object), then the message object only accepts remote frames. If MOCTR.DIR = 0 (receive object), then the message object only accepts data frames.
5. If MOAMR.MIDE = 1 in the Acceptance Mask Register, then the IDE bit of the received frame equals the MOAR.IDE bit in the Arbitration Register. IF MOAR.IDE = 1 then the message object only accepts frames with extended identifier. If MOAR.IDE = 0 then the message object only accepts standard frames. If MOAMR.MIDE = 0 then the IDE bit of the received frame is don't care, i.e. the message object accepts both standard and extended frames.
6. The identifier of the received frame matches the identifier stored in the Arbitration Register of the message object with respect to the acceptance mask in the MOAMR register. This means that each bit of the received identifier is equal to the corresponding identifier bit in the Acceptance Register, except those bits for which the corresponding mask bits in MOAMR are cleared. These identifier bits are don't care. **Figure 30-8** illustrates this identifier check.

A priority ordering relation is defined for the message objects:

A message object A has higher receive priority than a message object B if and only if the following conditions are fulfilled:

1. A belongs to a higher priority class than B, i.e. MOAR.PRI of A must be less than or equal to MOAR.PRI of B.
2. If both objects belong to the same priority class (PRI of A = PRI of B) then message object B is a list successor of A, i.e. B can be reached by means of successively stepping forward in the list, starting from A.

Among all messages that fulfill all 6 qualifying criteria the unique message object with highest receive priority wins acceptance filtering, i.e. is selected for storage of the received frame. All other message objects loose receive acceptance filtering.

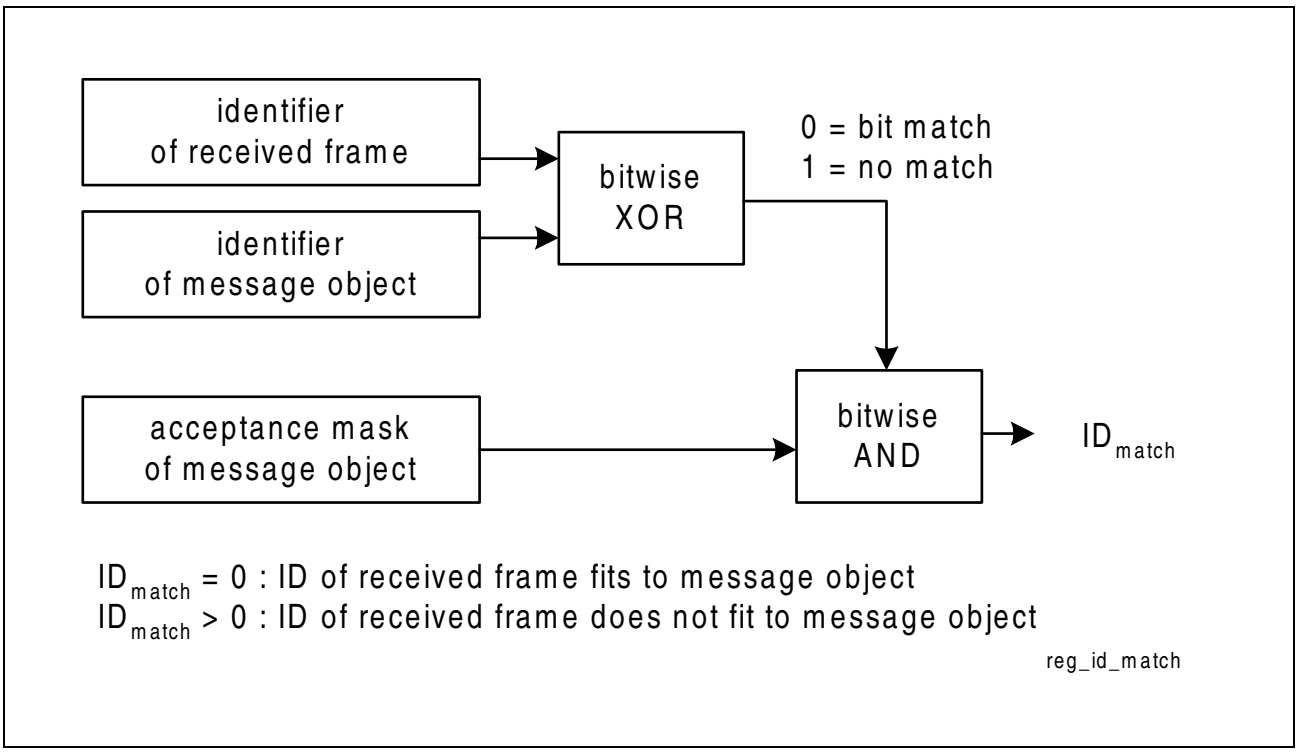


Figure 30-8 Received Message Identifier Acceptance Check

30.1.6.2 Transmit Acceptance Filtering

A message is requested for transmission by means of setting a transmit request in the message object which holds the message. If more than one message object has a valid transmit request for the same CAN node, then a single message object is chosen for actual transmission from the candidates, because only a single message object may be transmitted at the same time on a single CAN bus.

A message object qualifies for transmission on a given CAN node if and only if it meets the following criteria (**Figure 30-9**):

1. The message object is allocated to the list of the CAN node considered.
2. MSGVAL is set in the Message Object Control Register.
3. TXRQ is set in the Message Object Control Register.
4. TXEN0 and TXEN1 are set in the Message Object Control Register.

A priority order relation is defined for all qualifying objects to determine the message to be transmitted first: Let A and B be two message objects qualifying for transmission, where without loss of generality object B is assumed to be is a list successor of A, i.e. B can be reached by means of successively stepping forward in the list, starting from A. For both message objects associated CAN messages CAN_A and CAN_B are defined, where identifier, IDE and RTR bit are taken from MOAR.ID, MOAR.IDE and MOCTR.DIR.

If both message objects belong to a different priority class (different value of bit field MOAR.PRI), then the message object with lower MOAR.PRI value has higher transmit priority and will be transmitted first.

If both message objects belong to the same priority class (equal value of bit field MOAR.PRI), then message object A has higher transmit priority than object B if and only if one of the following conditions is fulfilled:

1. PRI = 10_B and CAN message CAN_A has higher or equal priority than CAN message CAN_B with respect to CAN arbitration rules (see **Table 30-12**).
2. PRI = 01_B or PRI = 11_B (priority by list order).

The unique message object that qualifies for transmission and has highest transmit priority wins transmit acceptance filtering, i.e. will be transmitted first. All other message objects lose the current transmit acceptance filtering round. They get a new chance in subsequent filtering rounds.

The priority rules 1 to 2 are valid for normal CAN operation.

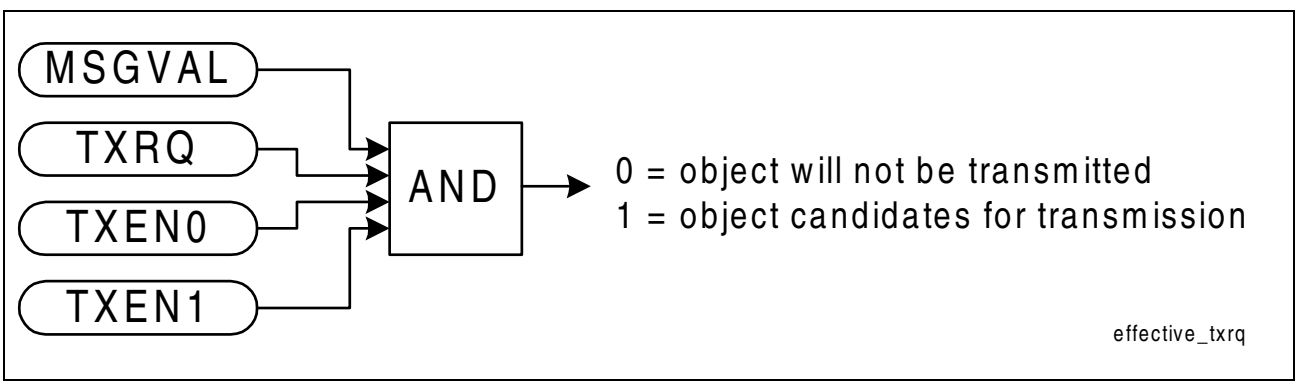


Figure 30-9 Effective Transmit Request of Message Object

30.1.7 Message Postprocessing Interface

When a message object has received or transmitted a frame successfully then the CPU may be notified to perform message postprocessing on the message object. The postprocessing interface of the MultiCAN module consists of two elements:

1. Message Interrupts to trigger postprocessing.
2. Message Pending Registers to aggregate the pending message interrupts into a common structure for postprocessing.

30.1.7.1 Message Interrupts

When the storage of a received frame into a message object or the successful transmission of a frame is completed then a message interrupt may be requested. For each message object both transmit and receive interrupts may be routed individually to one of the available interrupt nodes, as illustrated in **Figure 30-10**. A receive interrupt is not restricted to the direct storage of a received frame from the CAN node the message object belongs to. It also occurs upon frame storage induced by FIFO or gateway action. The TXPND and RXPND bits are set whenever a successful transmission/reception takes place, no matter if the respective interrupt is enabled or not.

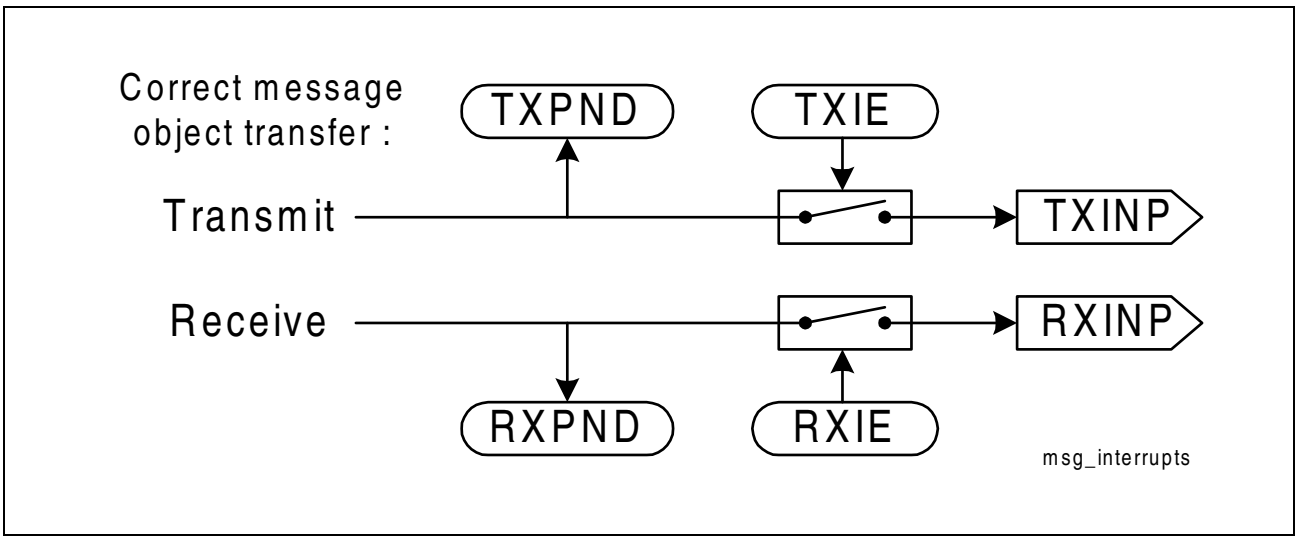


Figure 30-10 Message Interrupt Request Routing

30.1.7.2 Message Pending

When a message interrupt request is generated then also a message pending bit is set in one of the Message Pending Register. To this end the pending bit selection field MPN is defined in the Message Object Interrupt Pointer Register MOIPR. The value of MPN is combined with MOIPR.TXINP and MOIPR.RXINP to yield the effective bit position of the Pending bit, as illustrated in **Figure 30-11**. The bit position consists of 2 parts:

1. The high part (bits [7:5]) of the calculated position selects the Message Pending Register in which the pending bit will be set.
2. The low part (bits [4:0]) of the calculated position selects the position (0-31) of the pending bit within the 32-bit Message Pending Register.

The MCR.MPSEL bit field allows the inclusion of the interrupt request node pointer (RXINP for reception, TXINP for transmission) so as to implement different target location of the pending bit for receive and transmit.

The Message Pending Registers may be written by the CPU, but those bits that are written 1 are left unchanged and only those bits which are written 0 are cleared. This allows the clearing of individual bits with a single write access instead of a read/modify/write-back access. Thus there is no access conflict when the MultiCAN module sets another pending bit in the same register at the same time.

Each Message Pending Register is linked to an individual Message Index Register which displays the lowest bit position of all set (1) bits in the Message Pending Register. The Message Index Register is read-only and is updated immediately when the value of the corresponding Message Pending Register changes.

There is no direct link between the Message Pending Registers and the interrupt request nodes. Such a link may, however, be established by the application. For example, each interrupt request node could be linked to a unique Message Pending Register. The example shown in **Figure 30-12** links message Pending Register n to interrupt node n (n = 0-7).

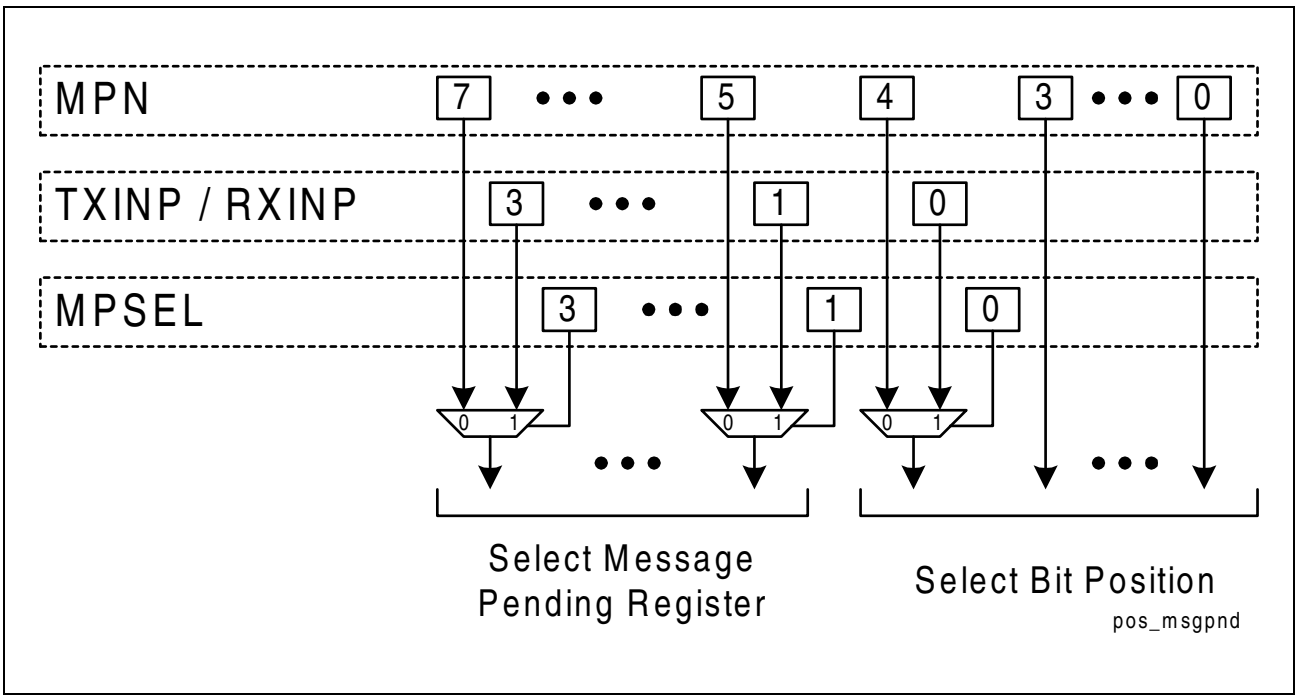


Figure 30-11 Target Location of the Message Pending Bit (Transmit/Receive)

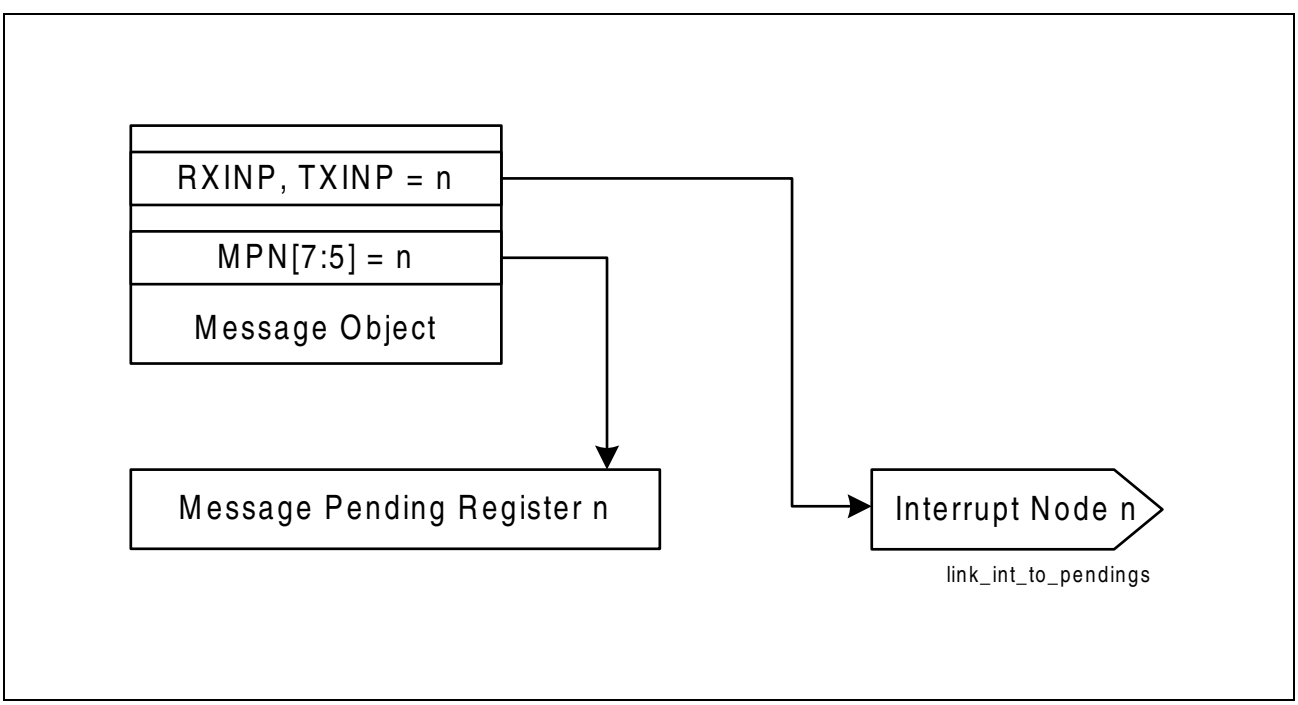


Figure 30-12 Example Link of Message Pending Registers to Interrupt Nodes

30.1.8 Message Object Data Handling

30.1.8.1 Frame Reception

When a message is received on the CAN bus then the storage of the message into a message object is prepared and performed according to the scheme shown in **Figure 30-13**. The MultiCAN module not just copies the received data into the message object, but it provides advanced features to enable consistent data exchange between MultiCAN and CPU.

MSGVAL

The MSGVAL (“Message Valid”) bit in the Message Object Control Register is the main switch of the message object. The MultiCAN module only stores information in the message object during the frame reception process when MSGVAL is set (MSGVAL = 1).

Whenever MSGVAL is reset (MSGVAL = 0) by the CPU then the MultiCAN module stops all ongoing write accesses to the message object so that the message object may be reconfigured by the CPU in subsequent write accesses to the message object without being disturbed by the MultiCAN.

RTSEL

When the CPU re-configures a message object (i.e. clears MSGVAL, modifies the message object and sets MSGVAL again) during CAN operation then the following scenario can occur:

1. The message object wins receive acceptance filtering.
2. The CPU clears MSGVAL to reconfigure the message object.
3. The CPU sets MSGVAL again after reconfiguration.
4. The end of the received frame is reached. As MSGVAL is set, the received data are stored in the message object, a message interrupt request is generated, gateway and fifo actions are processed etc.

The storage of the received data may be undesirable if the context of the message object has changed, because the old message object configuration has been used for acceptance filtering of the message.

Bit MOCTR.RTSEL (“Receive/Transmit Selected”) allows a message object to be disconnected from an ongoing frame reception:

When a message object wins receive acceptance filtering then bit RTSEL is set (RTSEL = 1) by the MultiCAN in order to indicate an upcoming frame delivery. The MultiCAN checks RTSEL for value 1 upon successful frame reception in order to verify that the object is still ready for receiving the frame. The received frame is stored in the message object (along with all subsequent actions such as message interrupts, FIFO & gateway actions, flag updates) only if RTSEL = 1.

When the user invalidates a message object during CAN operation (MSGVAL → 0) then the user should clear RTSEL before setting MSGVAL again (latest with the same write access that sets MSGVAL) in order to prevent the storage of a frame that belongs to the old context of the message object. Thus message object reconfiguration should consist of the following sequence of steps:

1. Clear MSGVAL.
2. Reconfigure message object while MSGVAL = 0.
3. Clear RTSEL and set MSGVAL.

RXEN

Bit MOCTR.RXEN enables a message object for frame reception. A message object can receive CAN messages from the CAN bus only if RXEN = 1. The MultiCAN evaluates RXEN only during receive acceptance filtering. After receive acceptance filtering RXEN is ignored, i.e. the value of RXEN has no influence on the actual storage of a received message in a message object.

Bit RXEN enables a “soft phase out” of a message object: When the user clears RXEN then a currently received CAN message for which the message object has won acceptance filtering is still stored in the message object, but for subsequent messages the message object no longer wins receive acceptance filtering.

RXUPD, NEWDAT and MSGLST

An ongoing frame storage process is indicated with the RXUPD (“Receive Updating”) bit in the Message Object Control Register MOCTR. The MultiCAN module sets RXUPD with the start and clears RXUPD with the end of a message object update (which consists of frame storage as well as flag updates).

After storing the received frame (identifier, IDE bit, DLC and for data frames also the data field) in the message object, MOCTR.NEWDAT (“New Data”) is set by the MultiCAN. If MOCTR.NEWDAT was already set then MOCTR.MSGLST (“Message Lost”) is also set in order to indicate data loss.

The RXUPD and NEWDAT flags may be used by the CPU to read consistent frame data from the message object during ongoing CAN operation. The following steps are recommended:

1. Clear NEWDAT
2. Read message content (identifier, data etc.) from message object
3. Read Message Object Control Register and check that both NEWDAT and RXUPD are cleared. If this is not the case, then go back to step 1.
4. As step 3 was successful, the read message content is consistent, i.e. has not been updated by the MultiCAN while reading.

The bits RXUPD, NEWDAT and MSGLST work in the same fashion for the reception of data as well as remote frames.

30.1.8.2 Frame Transmission

The process of message object transmission is illustrated in **Figure 30-14**. In addition to copying the message content (identifier, IDE bit, RTR = DIR bit, DLC and for data frames also the data field) to the internal transmit buffer of the CAN node that the message object belongs to, also several status flags are served and monitored in order to enable consistent data handling.

The transmission process (after transmit acceptance filtering) of a given message object makes no difference between remote and data frames.

MSGVAL, TXRQ, TXEN0, TXEN1

For the MSGVAL bit the section **"MSGVAL" on Page 30-24** for frame reception is also valid for transmission.

A message may only be transmitted if all four bits MSGVAL ("Message Valid"), TXRQ ("Transmit Request"), TXEN0 ("Transmit Enable 0"), TXEN1 ("Transmit Enable 1") of the Message Object Control Register are set (1) (see also **Figure 30-9**). Although these bits are equivalent with respect to the transmission process, they have different semantics:

Table 30-3 Bits to Set (1) in MOCTR for Message Transmission

Bit	Description
MSGVAL	**Message Valid** Main Switch of the Message Object
TXRQ	**Transmit Request** Standard Transmit Request bit. The CPU should set this bit whenever a message object shall be transmitted. TXRQ is cleared automatically at the end of the successful transmission, except when there are new data (indicated by NEWDAT = 1) to be transmitted. When the single transmit trial bit is set (STT = 1) in the Message Object Function Register then TXRQ is already cleared by the MultiCAN when the content of the message object is copied to the transmit frame buffer of the CAN node. A received remote request (i.e. remote frame received on CAN bus) sets bit TXRQ to request the transmission of the corresponding data frame.

Table 30-3 Bits to Set (1) in MOCTR for Message Transmission (cont'd)

Bit	Description
TXEN0	**Transmit Enable 0** This bit may be temporarily cleared by the CPU to suppress the transmission of this object when it writes new content to the data field. This avoids transmission of inconsistent frames which consist of a mixture of old and new data. Remote requests are still accepted during TXEN0 = 0, but transmission of the data frame is suspended until the CPU re-enables transmission (TXEN0 = 1).
TXEN1	**Transmit Enable 1** This bit is used in transmit FIFOs to select the message object which is transmit active within the FIFO structure. For message objects which are not transmit FIFO elements TXEN1 may either be set to 1 permanently or be used as a second, independent transmission enable bit.

RTSEL

When a message object has been identified to be transmitted next (by acceptance filtering) then the MultiCAN set bit MOCTR.RTSEL ("Receive/Transmit Selected").

When the MultiCAN copies the message object to the transmit buffer it checks bit RTSEL and the message is transmitted only if RTSEL = 1.

After the successful transmission of the message bit RTSEL is checked again and message postprocessing is only performed if RTSEL = 1.

A complete reconfiguration of an operating message object should be done by means of the following steps:

1. Clear MSGVAL ("Message Valid").
2. Reconfigure message object while MSGVAL = 0.
3. Clear RTSEL and set MSGVAL.

Here, clearing RTSEL ensures that the message object is disconnected from an ongoing/scheduled transmission and no message object processing (copying message to transmit buffer incl. clearing NEWDAT, clearing TXRQ, time stamp update, message interrupt etc.) within the old context of the object may occur after the message object becomes valid again, but within a new context.

NEWDAT

When the content of a message object has been transferred to the transmit buffer of the CAN node, bit NEWDAT ("New Data") is cleared to indicate that the transmit data of the message object are no longer new.

When the CAN transmission of the frame is successful and NEWDAT is still cleared (i.e. no new data have been copied to the message object in the meantime) then TXRQ (“Transmit Request”) is cleared automatically.

If the NEWDAT bit has been set again by the CPU (because a new frame is transmitted) then TXRQ is not cleared in order to enable the transmission of the new data.

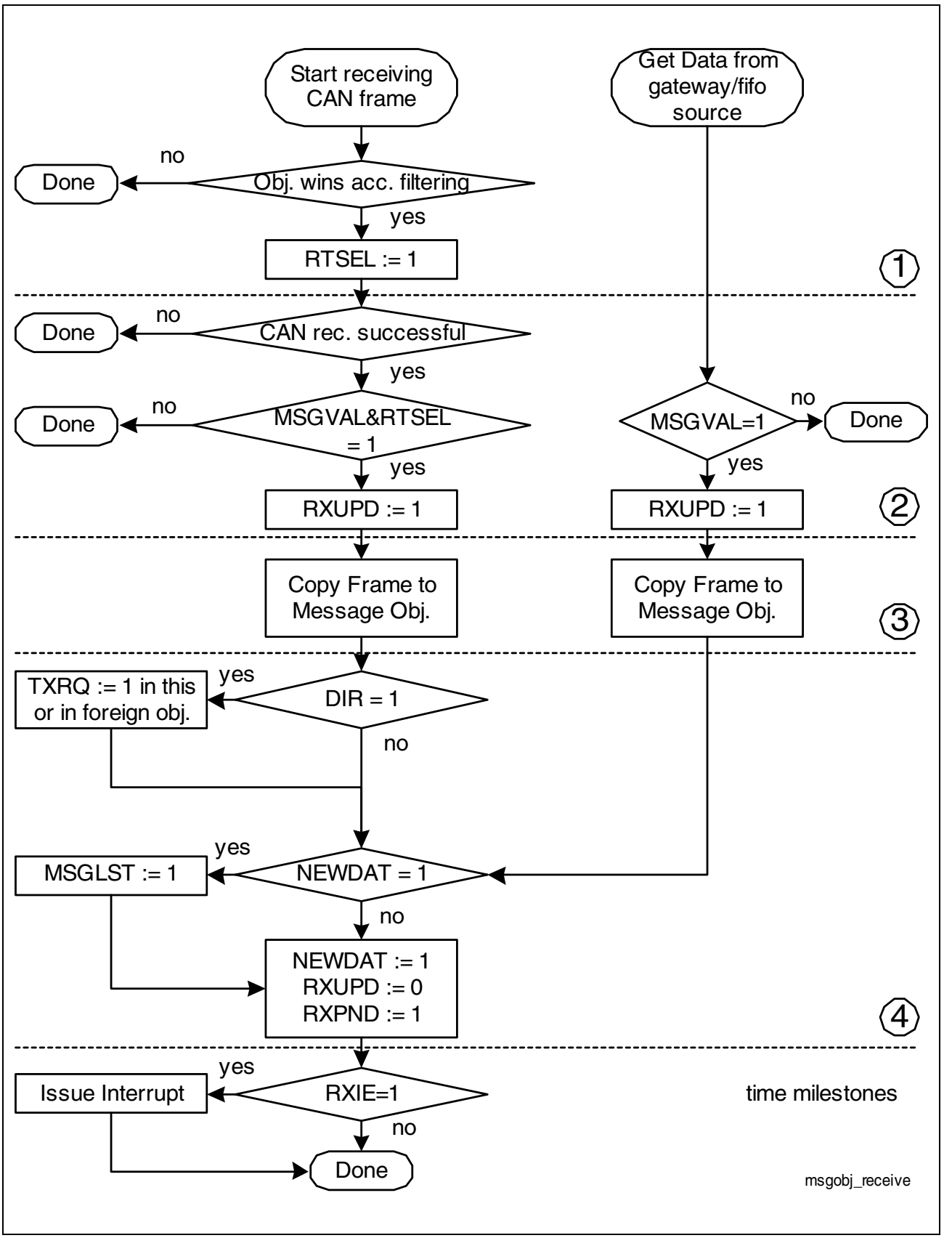


Figure 30-13 Data Delivery to Message Object by MultiCAN Module

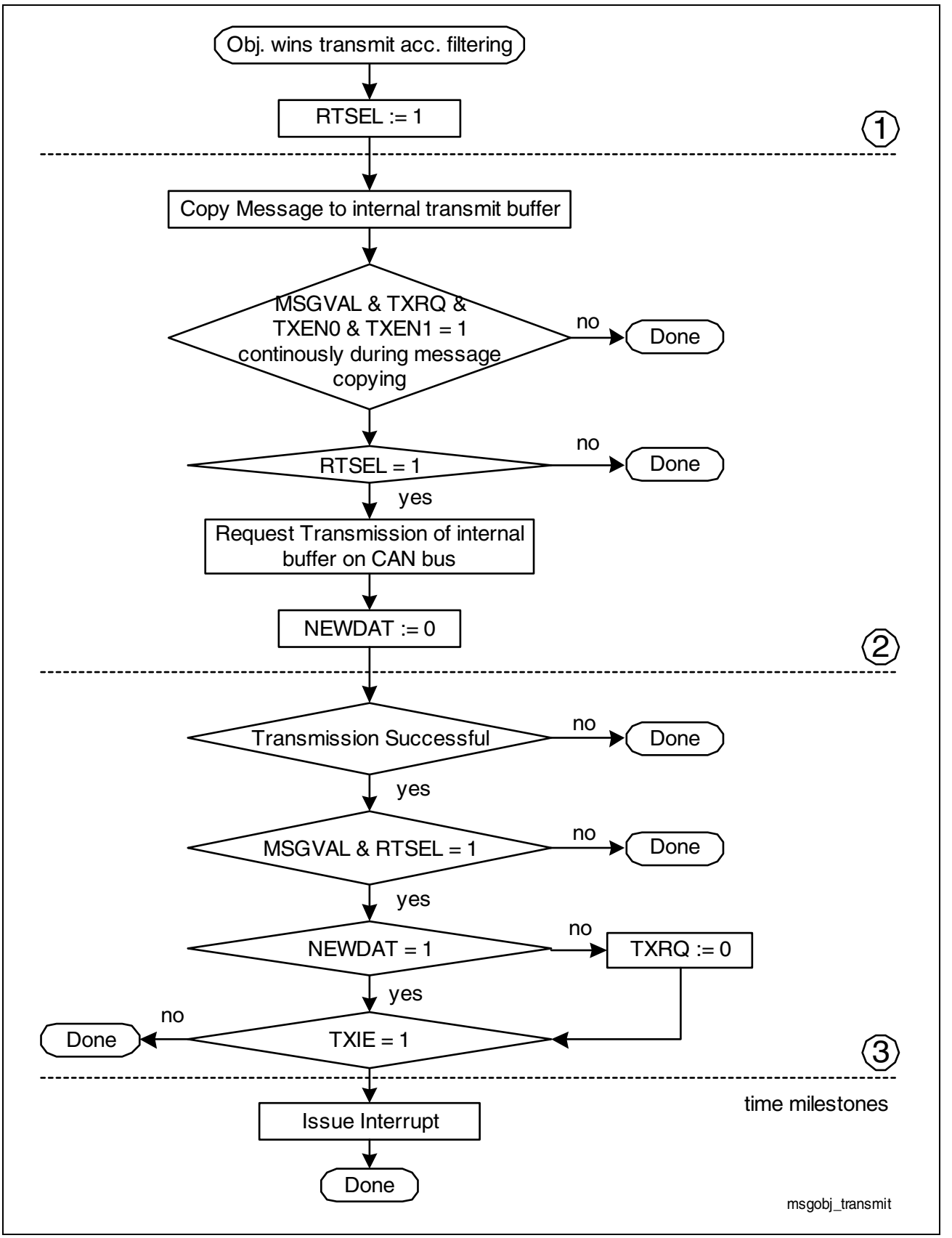


Figure 30-14 Transmission of a Message Object

30.1.9 Message Object Functionality

30.1.9.1 Standard Message Object Mode

Standard message mode is selected via MMC = 0000_B in the Message Object Function Control Register MOFCR. In this mode a message object may transmit and receive CAN frames according to the basic rules as described in the previous sections. Additional services such as Single Data Transfer Mode or Single Transmit Trial (see sections below) are available and may be selected individually by the user.

30.1.9.2 Single Data Transfer Mode

Single Data Transfer Mode is a useful feature in order to broadcast data over the CAN bus without unintended doubling of information. Single Data Transfer Mode is selected via bit MOFCR.SDT.

Message Reception

When a received message is stored in a message object and further messages are stored in the same message object before the CPU reads the first message object, then the content of the first message gets lost and is replaced with the content of the subsequent messages (indicated by MOCTR.MSGLST = 1).

If MOFCR.SDT = 1 (Single Data Transfer Mode activated) then the MultiCAN controller automatically clears the MSGVAL bit of the message object after the storage of a received data frame to prevent the reception of further messages.

The reception of a remote frame does not lead to the clearance of MSGVAL.

Message Transmission

When a message object receives a series of multiple remote requests, it transmits several data frames in response to the requests. If the data within the message object has not been updated in the time between the transmissions, the same data may be represented more than once on the CAN bus.

In Single Data Transfer Mode (MOFCR.SDT = 1), this is avoided because the MultiCAN controller automatically clears MSGVAL after the successful transmission of a data frame.

The transmission of a remote frame does not clear MSGVAL.

30.1.9.3 Single Transmit Trial

If the bit MOFCR.STT = 1, then the transmission request is cleared (MOCTR.TXRQ := 0) when the frame content of the message object has been copied to the internal transmit buffer of the CAN node. Thus the transmission of the message object is not tried again when it fails due to CAN bus errors.

30.1.9.4 Message Object FIFO Structure

In case of high CPU load it may be difficult to process a series of CAN frames in time. This may happen for the short term reception of multiple messages as well as the transmission of a series with tight due date.

Therefore a FIFO buffer structure has been implemented in order to avoid loss of incoming messages and to minimize the setup time for outgoing messages. The FIFO structure may also be used to automate the reception or transmission of a series of CAN messages and to generate a single message interrupt when the whole series is done.

There may be as many FIFOs in parallel as are required by the application. The number of FIFOs and their size are only limited by the number of message objects available. A FIFO may be installed, resized and deinstalled any time, even during CAN operation.

The basic structure of a FIFO is shown in **Figure 30-15**. A FIFO consists of a single base object (shown on the left side) and several slave objects (shown on the right side). The slave objects are chained together in the same list structure. The base object may be allocated to any list. Although **Figure 30-15** shows the base object as a separate item apart from the slave objects, it is also possible to integrate the base object at any place into the chain of slave objects, so that the base object is slave object, too (not possible for gateways). The FIFO structure fully relies on the list structure. The absolute object numbers of the message objects have no impact on the operation of the FIFO.

The base object needs not be allocated to the same list as the slave objects. Only the slave object must be allocated to a common list (as they are chained together). The BOT, CUR and TOP pointer link the base object to the slave objects, no matter weather the base object is allocated to the same or to another list than the slave objects.

The absolute minimum FIFO would consist of a single message object which is both FIFO base and FIFO slave (not very useful). The biggest possible FIFO would use all message objects of the MultiCAN module. Any sizes between these extremes are possible.

In the FIFO base object the boundaries of the FIFO are defined. The BOT field in the FIFO/Gateway Pointer Register of the base object points to the first (bottom) slave element in the FIFO. The TOP field in the FIFO/Gateway Pointer Register of the base object points to the last (top) slave element.

The CUR field in the FIFO/Gateway Pointer Register of the FIFO base object points to the actual slave object selected by the MultiCAN for message transfer. When a message transfer takes place with this object then CUR is moved to the next position. If CUR has already reached the top of the FIFO (CUR := TOP) then it is wrapped around to the bottom of the FIFO (CUR := BOT). Otherwise CUR is moved to the next message object in the list structure of the slave objects (CUR := PNEXT of current object). This scheme yields a circular FIFO structure where the fields BOT and TOP just establish the link from the last to the first element, which is missing in the linear structure.

The SEL field in the FIFO/Gateway Pointer register of the base object may be used for monitoring purposes. It allows the selection of any slave object and the generation of a message interrupt if the CUR pointer reaches the value of SEL. Thus SEL offers a convenient way to determine the end of a predefined series of message transfers, or it may be used to issue a warning to the CPU when the FIFO gets full.

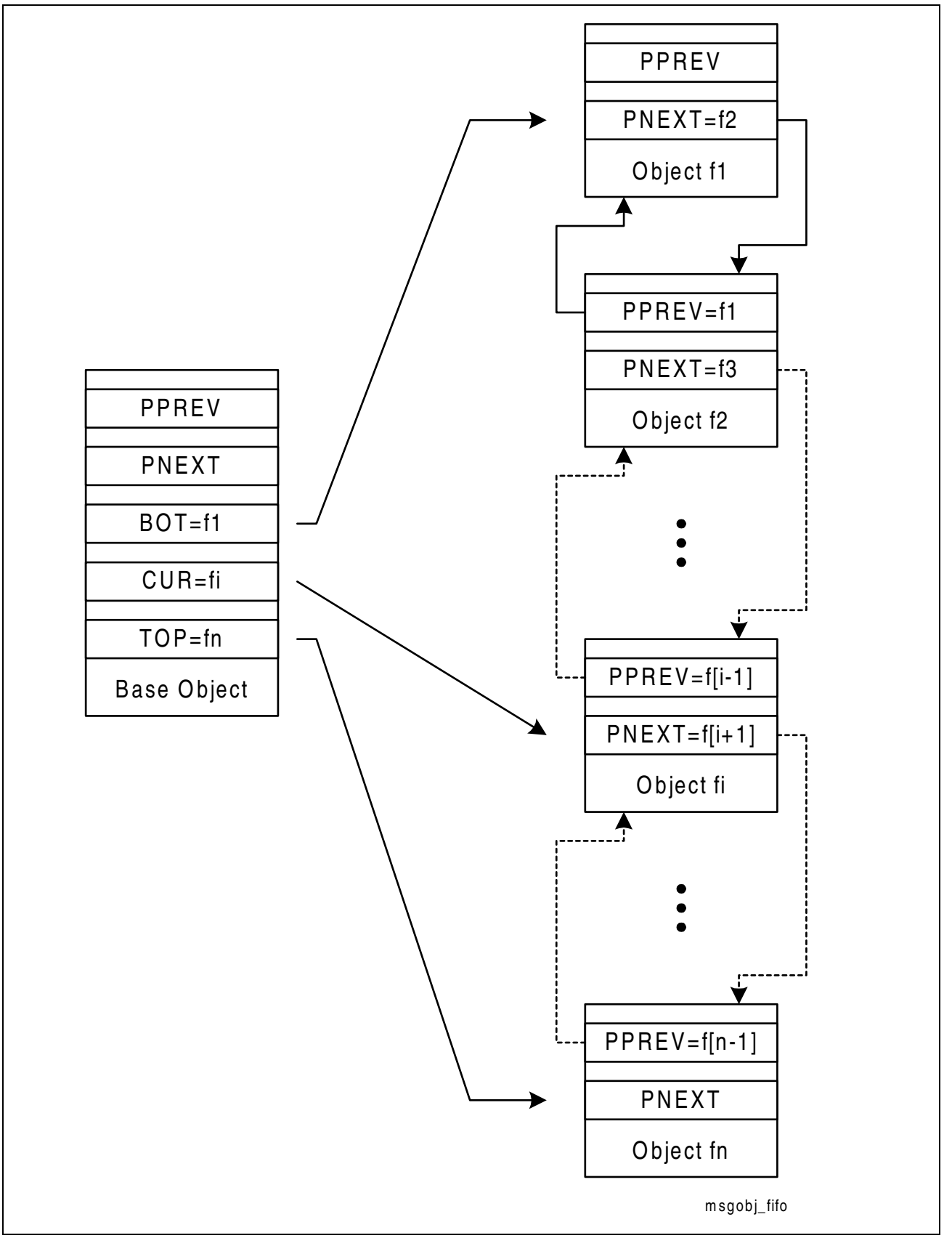


Figure 30-15 FIFO Structure with FIFO Base and n FIFO Destinations (Slaves)

30.1.9.5 Receive FIFO

The Receive FIFO structure is used to buffer incoming (received) remote or data frames.

A Receive FIFO is selected via MMC = 0001_B in the Message Object Function Control Register of the FIFO base object. This MMC code automatically designates the message object as FIFO base object. The message mode of the FIFO slave objects is not relevant for the operation of the Receive FIFO.

When the FIFO base object receives a frame from the CAN node it belongs to, then the frame is not stored in the base object. Instead the message is stored in the message object that is selected by the CUR pointer in the FIFO/Gateway Pointer Register of the FIFO base object.

The message object selected by CUR receives the CAN message as if it was the direct receiver of the message. However, MMC = 0000_B is implicitly assumed for the FIFO slave, i.e. a standard message delivery is performed. The actual message mode (MMC) of the FIFO slave is ignored. There is also no extra acceptance filtering to match the received frame against the identifier, IDE bit and DIR bit of the slave object.

When the FIFO base object receives a CAN frame then the MultiCAN moves the current pointer CUR to the next message object in the FIFO structure, which will then be used to store the next incoming message. The old value of CUR is used for the current transfer.

If bit OVIE is set in the Message Object Function Register of the FIFO base object and the pointer CUR reaches the value stored in SEL then a FIFO overflow interrupt request is generated. The interrupt request is generated on interrupt node TXINP (TXINP of the base object) immediately after the storage of the received frame into the slave object. Transmit interrupts are still generated if TXIE is set.

A CAN message is stored in a FIFO slave only if MSGVAL = 1 in both FIFO base and slave object.

30.1.9.6 Transmit FIFO

The Transmit FIFO structure is used to buffer a series of data or remote frames to be transmitted.

A Transmit FIFO base object is selected via MMC = 0010_B in the Message Object Function Control Register of the FIFO base object. Unlike the Receive FIFO the Transmit FIFO requires the explicit declaration of the FIFO slave objects via MMC = 0011_B. The CUR pointer of all slave objects must point back to the Transmit FIFO Base Object (to be initialized by user).

The TXEN1 bits of all message objects except the one which is selected by the CUR pointer of the base object must be cleared (to be initialized by user). TXEN1 of the message object selected by CUR must be set. CUR may be initialized to any FIFO slave object.

When tagging the message objects of the FIFO valid to start the operation of the FIFO then the base object must be tagged valid (MSGVAL := 1) first.

When a Transmit FIFO is deinstalled during operation, then the slave objects must be tagged invalid (MSGVAL := 0) first.

The Transmit FIFO uses the TXEN1 bit in the Message Object Control Register of all FIFO elements to select the actual message for transmission. Transmit acceptance filtering evaluates TXEN1 for each message object and a message object may win transit acceptance filtering only if TXEN1 is set. When a FIFO element has transmitted a message then in addition to standard transmit postprocessing (clear TXRQ, transmit interrupt etc.) the MultiCAN clears TXEN1 in that message object and moves the CUR pointer in the corresponding FIFO base object to the next message object to be transmitted. TXEN1 is set automatically in the next message object. Thus TXEN1 moves along the FIFO structure like a token to select the active element.

IF bit OVIE is set in the Message Object Function Register of the FIFO base object and the pointer CUR reaches the value stored in SEL then a FIFO overflow interrupt request is generated. The interrupt request is generated on interrupt node RXINP (RXINP of the base object) when postprocessing of the received frame is done. Receive interrupts are still generated for the Transmit FIFO base object if bit RXIE is set.

30.1.9.7 Gateway Mode

The Gateway Mode allows an automatic information transfer between two independent CAN bus systems to be established without CPU interaction.

The Gateway Mode operates on message object level. In Gateway mode, information is transferred between two message objects, resulting in an information transfer between the two CAN nodes to which the message objects are allocated. A gateway may be established between any pair of CAN nodes and there may be as many gateways as there are message objects available to build the gateway structure.

Gateway Mode is selected via MMC = 0100_B in the Message Object Function Control Register of the gateway source object. The gateway destination object is selected by the CUR pointer in the FIFO/Gateway Pointer Register of the source object. The gateway destination object just needs to be valid (MSGVAL = 1), all other settings are not relevant for the information transfer from the source object to the destination object.

A gateway source object behaves like a standard message objects, but when a CAN frame has been received and stored in the source object, some additional actions are performed by the MultiCAN (**Figure 30-16**):

1. If bit DLCC is set in the Message Object Function Control Register of the source object, then the DLC code is copied from the source object to the destination object.
2. If bit IDC is set in the Message Object Function Control Register of the source object, then the identifier and the IDE bit are copied from the source object to the destination object.
3. If bit DATC is set in the Message Object Function Control Register of the source object, then the data field is copied from the source object to the destination object.
4. If bit GDFS is set in the Message Object Function Control Register of the source object, then TXRQ is set in the Message Object Control Register of the destination object.
5. Bit RXPND and Bit NEWDAT are set in the Message Object Control Register of the destination object.
6. A message interrupt request is generated for the destination object if RXIE is set in the Message Object Control Register of the destination object.
7. The current pointer CUR in the FIFO/Gateway Pointer Register of the source object is moved to the next destination object according to the FIFO rules as described in **Section 30.1.9.4**. A gateway with a single (static) destination object is obtained by means of setting TOP = BOT = CUR = destination object.

The link from the source to the destination object works in the same way as the link from a FIFO source to a FIFO slave. This means that a gateway with an integrated destination FIFO may be created (**Figure 30-15**), where the object on the left in **Figure 30-15** is the gateway source object and the message objects on the right side are the gateway destination objects.

The gateway works in the same way for the reception of data frames (source object is receive object, i.e. DIR = 0) as well as for the reception of remote frames (source object is transmit object).

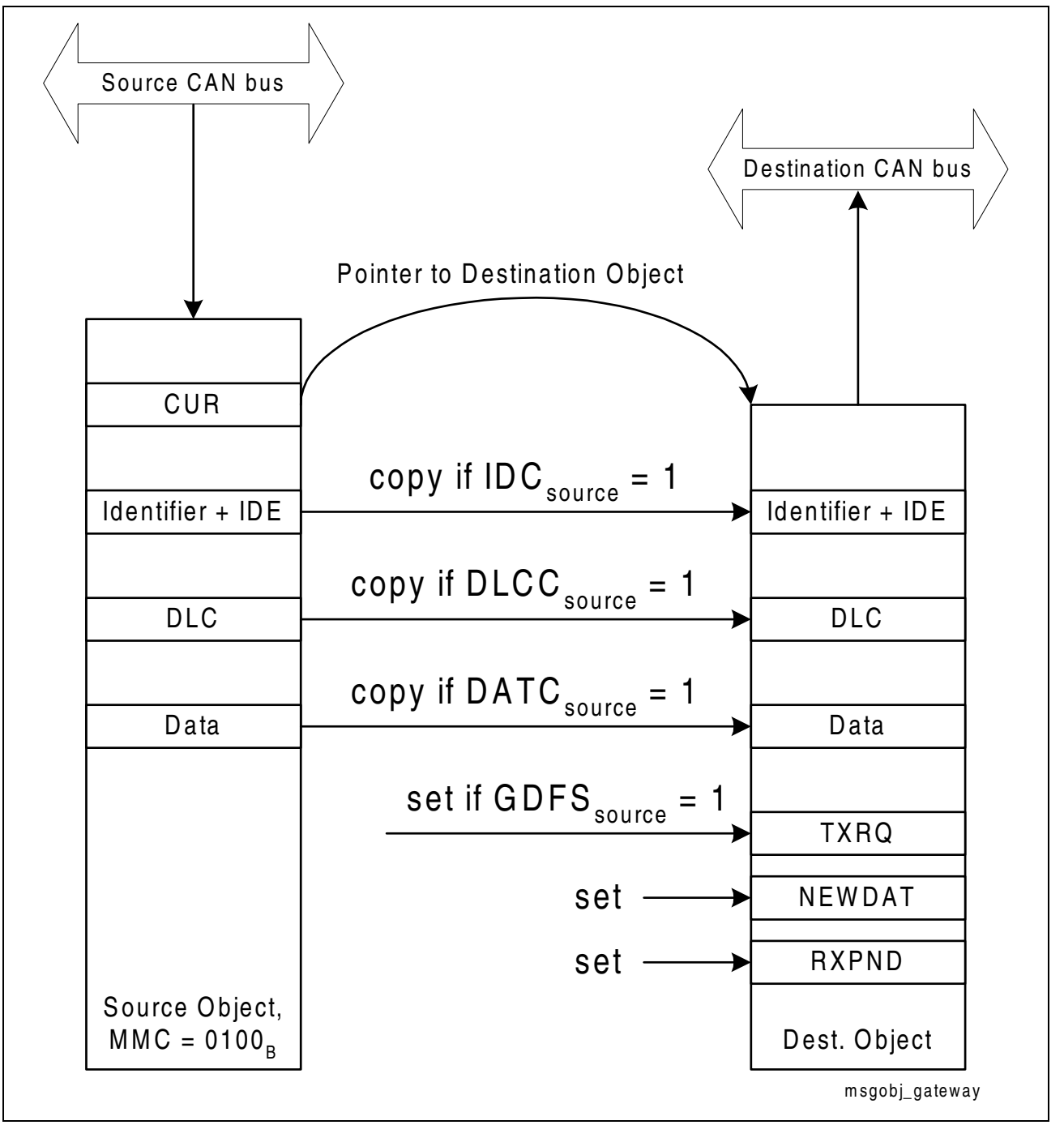


Figure 30-16 Gateway Transfer from Source to Destination

30.1.9.8 Foreign Remote Requests

When a remote frame received on a CAN node is stored in a message object, then a transmit request is set in order to trigger the answer (data frame transmission) to the request or to automatically issue a secondary request. If bit MOFCR.FRREN is cleared (FRREN = 0) of the message object where the remote request is stored, then MOCTR.TXRQ is set of the same message object.

If bit MOFCR.FRREN is set (FRREN = 1: foreign remote request enabled) then MOCTR.TXRQ is set in the message object that is referenced by pointer CUR in the FIFO/Gateway Pointer Register. The value of CUR is, however, not changed by this feature.

Although the foreign remote request feature works independently from the selected message mode, it is especially useful for gateways to issue a remote request on the source of a gateway upon the reception of a remote request on the gateway destination. According to the setting of FRREN in the gateway destination object there are two ways to handle remote requests that appear on the destination side (assuming that the source object is a receive object and the destination object is a transmit object, i.e. DIR_{source} = 0 and $DIR_{destination}$ = 1):

FRREN = 0 in the Gateway Destination Object

1. A remote frame is received by gateway destination.
2. TXRQ is set automatically in the gateway destination object.
3. A data frame with the current data stored in the destination object is transmitted on the destination bus.

FRREN = 1 in the Gateway Destination Object

1. A remote frame is received by gateway destination.
2. TXRQ is set automatically in the gateway source object (must be referenced by CUR pointer of the destination object).
3. A remote request is transmitted by the source object (which is a receive object) on the source CAN bus.
4. The receiver of the remote request responds with a data frame on the source bus.
5. The data frame is stored in the source object.
6. The data frame is copied to the destination object (gateway action).
7. TXRQ is set in the destination object (assuming $GDFS_{source}$ = 1).
8. The new data stored in the destination object is transmitted on the destination bus, as response to the initial remote request on the destination bus.

30.2 MultiCAN Kernel Registers

The register set of the MultiCAN module consists of three distinct subsets:

1. The **Global Module Registers** apply to the whole MultiCAN module and exist only once.
2. The **CAN Node Registers** apply to a single CAN node and thus exist once for each CAN node.
3. The **Message Object Registers** define the message objects and thus exist once for each message object.

Table 30-4 shows the address map of the MultiCAN module with respect to the base address of the MultiCAN module.

Table 30-4 MultiCAN Address Map (Relative to MultiCAN Base Address)

Register Group	Start Address	Total Range
Global Module Registers	$+100_H$	$+100_H$ to $+1FF_H$
CAN Node Registers for CAN node x = 0-3	$+200_H + 100_H \times x$	$+200_H$ to $+5FF_H$
Message Objects n = 0-127	$+600_H + 20_H \times n$	$+600_H$ to $+15FF_H$

Figure 30-17 and **Table 30-5** show all registers associated with the CAN Kernel.

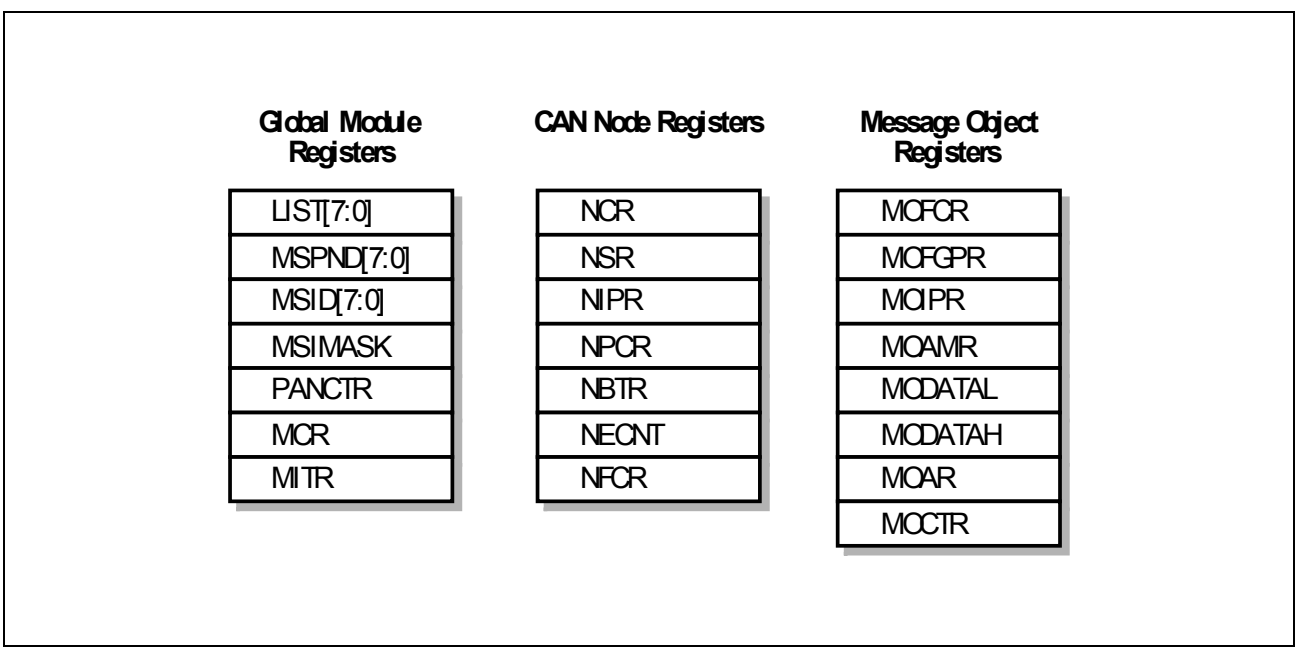


Figure 30-17 MultiCAN Kernel Registers

Note: The CAN RAM is automatically initialized after reset by the list controller in order to ensure correct list pointers in each message object. The end of this CAN RAM initialization is indicated by bit PANCTR.BUSY becoming inactive.

Table 30-5 CAN Kernel Registers

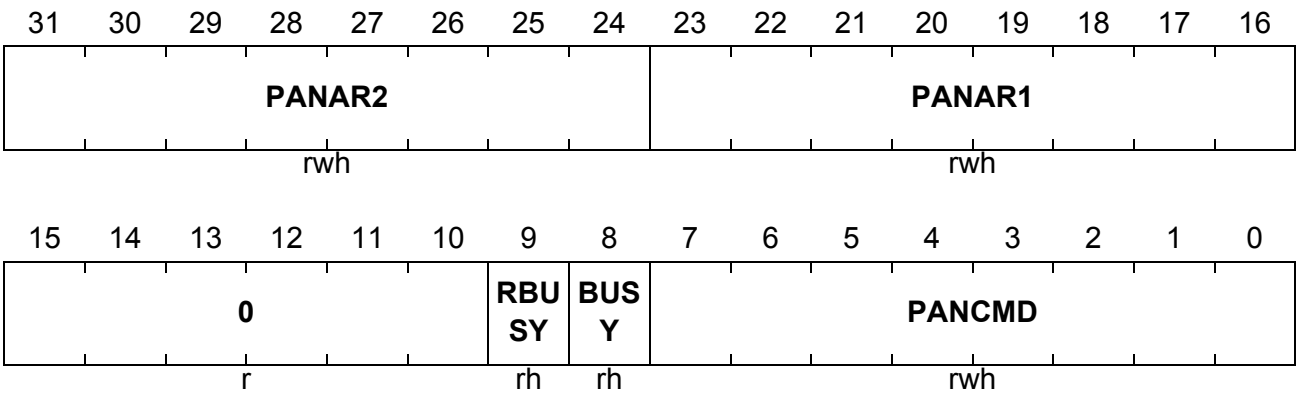
Register Short Name	Register Long Name	Offset Address	Description see
Global Module Registers			
LIST[7:0]	List Registers 0-7	0000_H to $001C_H$	Page 30-48
MSPND[7:0]	Message Pending Registers 0-7	0020_H to $003C_H$	Page 30-49
MSID[7:0]	Message Index Registers 0-7	0040_H to $005C_H$	Page 30-49
MSIMASK	Message Index Mask Register	$00C0_H$	Page 30-50
PANCTR	Panel Control Register	$00C4_H$	Page 30-42
MCR	Module Control Register	$00C8_H$	Page 30-45
MITR	Module Interrupt Trigger Register	$00CC_H$	Page 30-46
CAN Node Registers			
NCR	CAN Node Control Register	0000_H	Page 30-51
NSR	CAN Node Status Register	0004_H	Page 30-55
NIPR	CAN Node Interrupt Pointer Register	0008_H	Page 30-59
NPCR	CAN Node Port Control Register	000C	Page 30-60
NBTR	CAN Node Bit Timing Register	0010_H	Page 30-61
NECNT	CAN Node Error Counter Register	0014_H	Page 30-63
NFCR	CAN Node Frame Counter Register	0018_H	Page 30-60
Message Object Registers			
MOFCR	Message Object Function Control Register	0000_H	Page 30-74
MOFGPR	Message Object FIFO/Gateway Pointer Register	0004_H	Page 30-78
MOIPR	Message Object Interrupt Pointer Register	0008_H	Page 30-73
MOAMR	Message Object Acceptance Mask Register	$000C_H$	Page 30-79
MODATAL	Message Object Data Register Low	0010_H	Page 30-83
MODATAH	Message Object Data Register High	0014_H	Page 30-84
MOAR	Message Object Arbitration Register	0018_H	Page 30-80
MOCTR	Message Object Control Register	$001C_H$	Page 30-68

30.2.1 Global Module Registers

All list operations such as allocation, deallocation and relocation of message objects within the list structure are performed via the Command Panel. It is not possible to modify the list structure directly by means of writing to the message objects and the LIST registers.

A new command is started by means of writing the command arguments and the command code to the PANCTR register.

PANCTR
Panel Control Register **Reset Value: 0000 0301$_H$**



Field	Bits	Type	Description
PANCMD	[7:0]	rwh	**Panel Command** A new command is started by means of writing the command number to PANCMD. At the end of a panel command the NOP (no operation) command code is automatically written to PANCMD.
BUSY	8	rh	**Panel Busy** 0 Panel has finished command and is ready to accept a new command. 1 Panel operation is in progress.
RBUSY	9	rh	**Result Busy** 0 No update of PANAR1 and PANAR2 is scheduled by the list controller. 1 A list command is running (BUSY = 1) that will write results to PANAR1 and PANAR2, but the results are not yet available.
PANAR1	[23:16]	rwh	**Panel Argument 1**
PANAR2	[31:24]	rwh	**Panel Argument 2**
0	[15:10]	r	**Reserved;** read as 0; should be written with 0.

A panel operation consists of a command code to be written to PANCMD and up to 2 panel arguments (PANAR1, PANAR2). Commands that have a return value deliver it to the PANAR1 field. Commands that deliver an error flag post it to bit 31 of the PANCTR register (i.e. bit 7 of PANAR2).

Table 30-6 Panel Commands

Code	PANAR2	PANAR1	Command Description
0	–	–	**No Operation** Writing value 0 to PANCMD has no effect. No new command is started.
1	**Result:** Bit 7: ERR Bit 6-0: undefined	–	**Initialize Lists** Run the initialization sequence to reset the CTRL and LIST field of all message objects and the list registers LIST[7:0] to their reset values. This results in the deallocation of all message objects. The initialization command requires that bits INIT and CCE are set in the Node Control Register of all CAN nodes 0-3. An ERR bit (bit 7 of PANAR2) reports the success of the operation: 0 Success 1 Not all INIT and CCE bits are set. Thus no initialization is performed The initialization command is automatically performed with each reset of the MultiCAN module, but with the exception that all message object registers are reset.
2	**Argument:** List Index	**Argument:** Message Object Number	**Static Allocate** Allocate a given message object to a list. The message object is removed from the list that it currently belongs to and appended to the end of the list that is given by PANAR2. This command is also used to deallocate a message object. In this case the target list is the list of unallocated elements. (PANAR2 = 0).

Table 30-6 Panel Commands (cont'd)

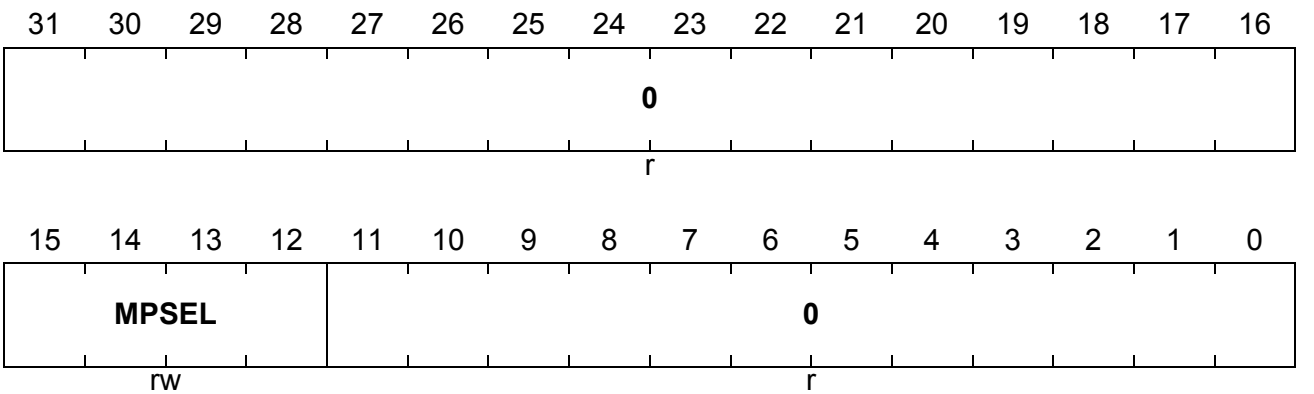
Code	PANAR2	PANAR1	Command Description
3	**Argument:** List Index **Result:** Bit 7: ERR Bit 6-0: undefined	**Result:** Message Object Number	**Dynamic Allocate** Allocate the first message object of the list of unallocated objects to the selected list. The message object is appended to the end of the list. The message number of the message object is returned in PANAR1. An ERR bit (bit 7 of PANAR2) reports the success of the operation: 0 Success 1 The operation has not been performed because the list of unallocated elements was empty
4	**Argument:** Destination Object Number	**Argument:** Source Object Number	**Static Insert Before** Remove a message object (source object) from the list that it currently belongs to and insert it before a given destination object into the list structure of the destination object. The source object thus becomes the predecessor of the destination object.
5	**Argument:** Destination Object Number **Result:** Bit 7: ERR Bit 6-0: undefined	**Result:** Object Number of inserted object	**Dynamic Insert Before** Insert a new message object before a given destination object. The new object is taken from the list of unallocated elements (the first element is chosen). The number of the new object is delivered as result to PANAR1. An ERR bit (bit 7 of PANAR2) reports the success of the operation: 0 Success 1 The operation has not been performed because the list of unallocated elements was empty
6	**Argument:** Destination Object Number	**Argument:** Source Object Number	**Static Insert Behind** Remove a message object (source object) from the list that it currently belongs to and insert it behind a given destination object into the list structure of the destination object. The source object thus becomes the successor of the destination object.

Table 30-6 Panel Commands (cont'd)

Code	PANAR2	PANAR1	Command Description
7	**Argument:** Destination Object Number **Result:** Bit 7: ERR Bit 6-0: undefined	**Result:** Object Number of inserted object	**Dynamic Insert Behind** Insert a new message object behind a given destination object. The new object is taken from the list of unallocated elements (the first element is chosen). The number of the new object is delivered as result to PANAR1. An ERR bit (bit 7 of PANAR2) reports the success of the operation: 0 Success 1 The operation has not been performed because the list of unallocated elements was empty
8 - 255	–	–	**Reserved**

The MCR register contains basic settings to define the operation of the module.

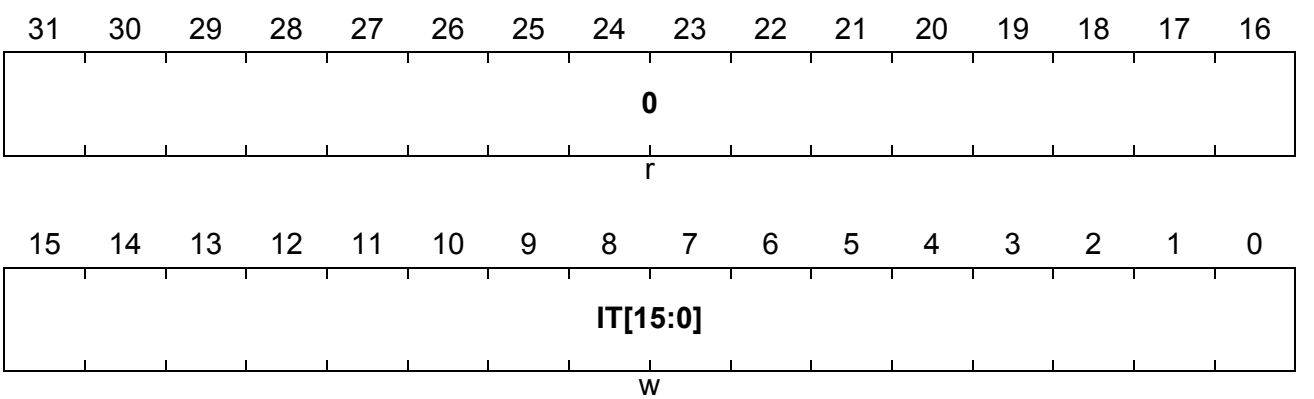
MCR
Module Control Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
MPSEL	[15:12]	rw	**Message Pending Selector** MPSEL allows the calculation of the bit position of the message pending bit, which is set after a message reception/transmission interrupt from a mixture of RXINP, TXINP, and MPN (Message Pending Number). With the definitions: INP … RXINP upon message reception, TXINP upon message transmission MPN … 8-bit message pending number the effective position of the message pending bit is calculated according to the formula: POS = ((INP & MPSEL) << 4) | ((MPN & (~MPSEL << 4)) | (MPN & $0F_H$) If MPSEL = 0 then the position is simply given by the message pending number MPN. If MPSEL = 1111_B then the upper 4 bits of the position is given by the interrupt node pointer INP and the lower 4 bits are taken from MPN.
0	[11:0], [31:16]	r	**Reserved;** read as 0; should be written with 0.

The ITR register allows interrupt requests to be triggered on each interrupt node by software.

MITR
Module Interrupt Trigger Register **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
IT[15:0]	[15:0]	w	**Interrupt Trigger** Writing value 1 to bit n (n = 15-0) generates an interrupt request on interrupt node n. Writing value 0 has no effect. Reading delivers always 0. More than one interrupt request may be generated at the same time by means of writing 1 to several bit positions of IT with a single write access.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

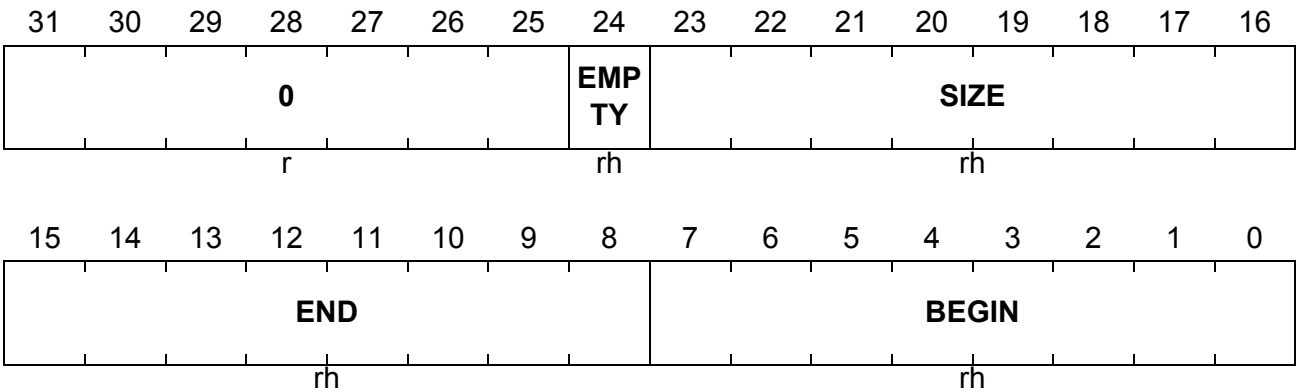
Each CAN node has an own list which defines the message objects that are allocated to the respective node. In addition to that there is the list of all unallocated objects and a general purpose user list which is not associated to a CAN node. Each list is assigned a list index according to **Table 30-7**.

Table 30-7 List Indices

List Index	Description
0	List of unallocated elements
1 to n_nodes	Lists associated to a CAN node. List index i belongs to CAN node i+1.
n_nodes + 1 to n_lists - 1	Free user lists, which are not associated to a CAN node.

Each list is terminated with a List Register which defines the first and the last element in the list.

LIST0
List Register 0 **Reset Value: 007F 7F00$_H$**
LISTi (i = 1-7)
List Register i **Reset Value: 0100 0000$_H$**

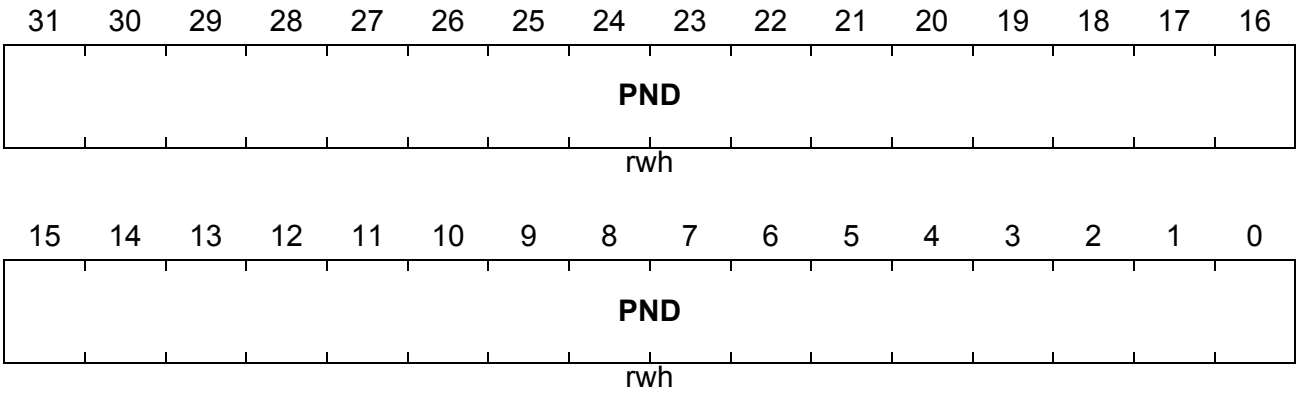


Field	Bits	Type	Description
BEGIN	[7:0]	rh	**List Begin** Pointer to the first message object in the list i.
END	[15:8]	rh	**END Pointer** Pointer to the last message object in the list i.
SIZE	[23:16]	rh	**Size of List** The number of elements in the list I is given by #elements = SIZE + 1, provided the list is not empty. If the list i is empty, the value of SIZE is zero.
EMPTY	24	rh	**List Empty Indication** 0 At least one message object is allocated to list i. 1 No message object is allocated to the list i.
0	[31:25]	r	**Reserved;** read as 0; should be written with 0.

When a message object generates an interrupt request upon the transmission or reception of a message, then the request is routed to the interrupt node selected by TXINP or RXINP of the message object. As there are more message objects than interrupt nodes, an interrupt routine typically processes requests from more than one message object. Therefore a priority selection mechanism is implemented in the MultiCAN module to select the highest priority object within a collection of message objects.

The MSPNDK register contains the interrupt pending.

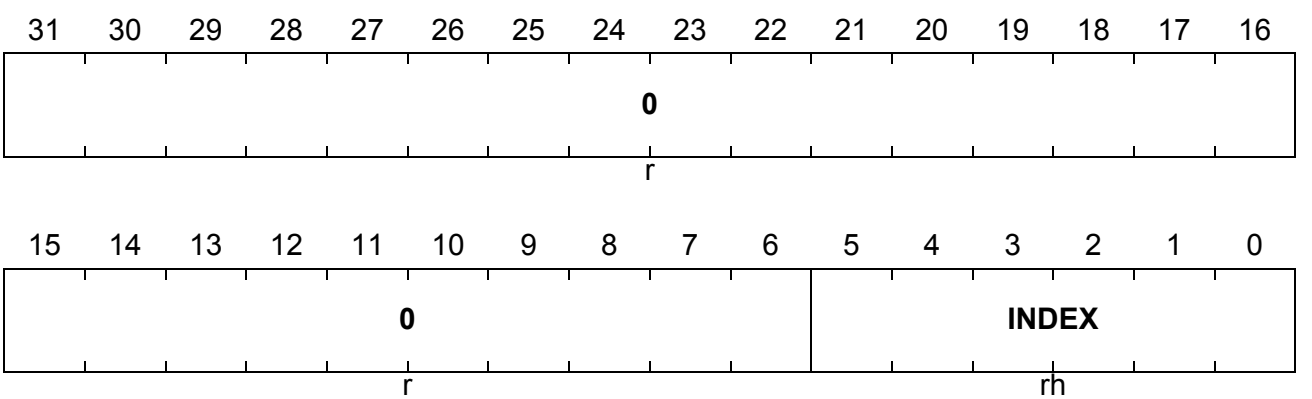
MSPNDk (k = 0-7)
Message Pending Register k **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
PND	[31:0]	rwh	**Message Pending** When a message interrupt occurs then the message object sets a bit in one of the MSPND register, where the bit position is given by the MPN[4:0] field of the MOIPR register of the message object. The register selection n is given by the higher bits of MPN. The register bits may be cleared by software (write 0), but writing 1 has no effect.

Each Message Pending Register has a Message Index Register associated to it. The Message Index Register shows the active (set) pending bit with lowest bit position within groups of pending bits.

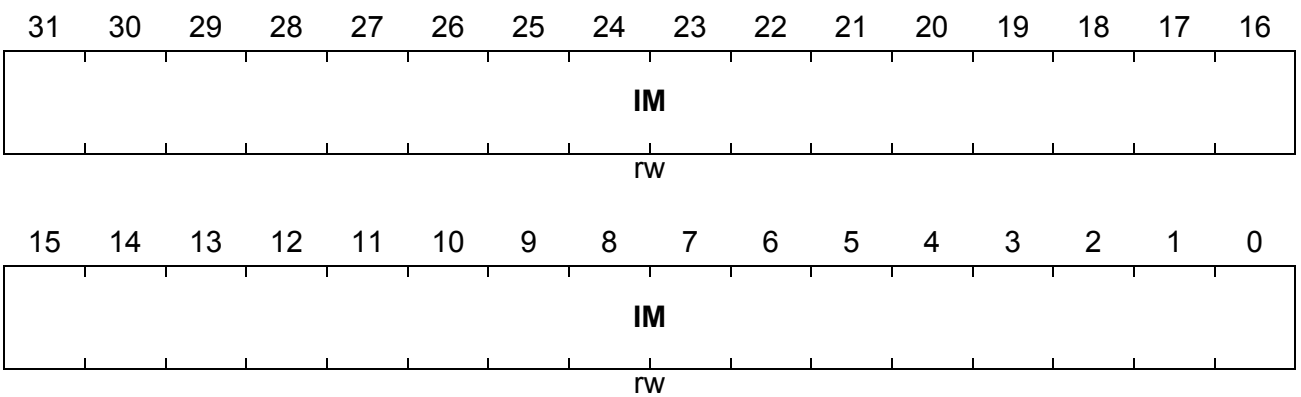
MSIDk (k = 0-7)
Message Index Register k **Reset Value: 0000 0020$_H$**



Field	Bits	Type	Description
INDEX	[5:0]	rh	**Message Pending Index** The value of INDEX is given by the bit position i of the pending bit of MSPNDk with the following properties: 1. MSPNDk[i] & IM[i] = 1 2. i = 0 or MSPNDk[i-1:0] & IM[i-1:0] = 0 If no bit of MSPNDk satisfies these conditions then INDEX reads 100000_B. Thus INDEX shows the position of the first pending bit of MSPNDk, where only those bits of MSPNDk which are selected in the Message Index Mask Register are taken into account.
0	[31:6]	r	**Reserved;** read as 0; should be written with 0.

The Message Index Mask Register selects individual bits for the calculation of the Message Pending Index. The Message Index Mask Register is used commonly for all Message Pending registers and their associated Message Index registers.

MSIMASK
Message Index Mask Register **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
IM	[31:0]	rw	**Message Index Mask** Only those bits in MSPNDk for which the corresponding Index Mask bits are set contribute to the calculation of the Message Index.

30.2.2 CAN Node Registers

The CAN node registers exist once for each CAN node of the MultiCAN module. They contain information that is directly related to the operation of the CAN nodes and which may not be shared among the nodes.

The NCR register contains basic settings that define the operation of the CAN node and the interaction of the CAN node with the message objects.

NCRx (x = 0-3)
Node x Control Register **Reset Value: 0000 0001$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							SUS EN	CAL M	CCE	0	CAN DIS	ALIE	LECI E	TRIE	INIT
r							rw	rw	rw	r	rw	rw	rw	rw	rwh

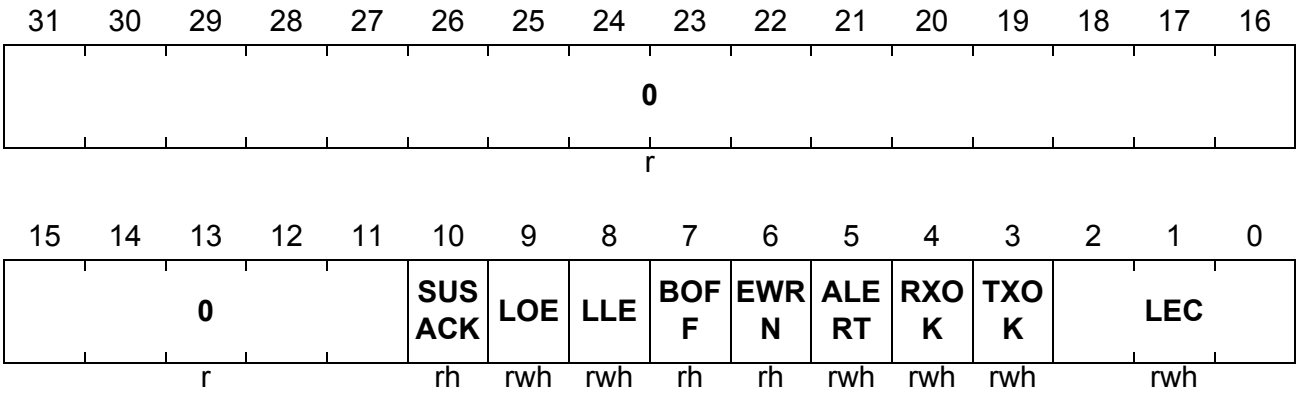
Field	Bits	Type	Description
INIT	0	rwh	**Node Initialization** 0 Resetting bit INIT enables the participation of the node in the CAN traffic. If the CAN node is in the bus off state then the ongoing bus off recovery (which does not depend on the INIT bit) is continued. With the end of the bus off recovery sequence the CAN node is allowed to take part in the CAN traffic. If the CAN node is not in the bus off state a sequence of 11 consecutive recessive bits must be detected before the node is allowed to take part in the CAN traffic. 1 Setting this bit terminates the participation of this node in the CAN traffic. Any ongoing frame transfer is cancelled and the transmit line goes **recessive**. If the CAN node is in the bus off state then the running bus off recovery sequence is continued. If the INIT bit is still set after the successful completion of the bus off recovery sequence, i.e. after detecting 128 sequences of 11 consecutive **recessive** bits (11 × 1) then the CAN node leaves the bus off state but remains inactive as long as INIT remains set. Bit INIT is automatically set when the CAN node becomes ‘bus off’.
TRIE	1	rw	**Transfer Interrupt Enable** If this bit is set, then an interrupt request is generated upon the successful reception or transmission of a CAN frame. The interrupt node is selected by TRINP in the CAN Node Interrupt Pointer Register.
LECIE	2	rw	**LEC Indicated Error Interrupt Enable** If this bit is set, then an interrupt request is generated upon each update of the LEC field in the Node Status Register leading to LEC > 0 (CAN protocol error). The interrupt node is selected by LECINP in the CAN Node Interrupt Pointer Register.

Field	Bits	Type	Description
ALIE	3	rw	**Alert Interrupt Enable** If this bit is set then an alert interrupt is generated on one of the following events: 1. A change of bit BOFF in the CAN Node Status Register. 2. A change of bit EWRN in the CAN Node Status Register. 3. A List Length Error, which also sets bit LLE in the CAN Node Status Register. 4. A List Object Error, which also sets bit LOE in the CAN Node Status Register. 5. Bit INIT has been set by the MultiCAN. The interrupt is requested on the interrupt node selected by ALINP in the CAN Node Interrupt Pointer Register.
CANDIS	4	rw	**CAN Disable** Setting this bit disables the CAN node. The CAN node first waits until it is BUS IDLE or BUS OFF. Then bit INIT is automatically set and an alert interrupt is generated if bit ALIE is set.
CCE	6	rw	**Configuration Change Enable** 0 The Bit Timing Register, the Port Control Register and the Error Counter Register may only be read. All attempts to modify them are ignored. 1 The Bit Timing Register, the Port Control Register and the Error Counter Register may be read and written.
CALM	7	rw	**Can Analyze Mode** If this bit is set then the CAN node operates in analyze mode. This means that messages may be received, but not transmitted. No acknowledge is sent on the CAN bus upon frame reception. Active error flags are sent **recessive** instead of **dominant**. The transmit line is continuously held at **recessive** (1) level. Bit CALM can be written only while bit INIT is set.

Field	Bits	Type	Description
SUSEN	8	rw	**Suspend Enable** This bit allows the setting of the CAN node into suspend mode via OCDS (on-chip debug support): 0 An OCDS suspend trigger is ignored by the CAN node. 1 An OCDS suspend trigger disables the CAN node: As soon as the CAN node becomes BUS IDLE or BUS OFF bit INIT is internally forced to 1 to disable the CAN node. The actual value of bit INIT remains unchanged. Bit SUSEN is reset via OCDS Reset.
0	5, [31:9]	r	**Reserved;** read as 0; should be written with 0.

The NSR register reports errors as well as successfully transferred CAN frames.

NSRx (x = 0-3)
Node x Status Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
LEC	[2:0]	rwh	**Last Error Code** The encoding of this bit field is detailed in **Table 30-8**.
TXOK	3	rwh	**Message Transmitted Successfully** 0 No successful transmission since last flag reset. 1 A message has been transmitted successfully (error free and acknowledged by at least another node). TXOK must be reset by software (write 0). Writing 1 has no effect.
RXOK	4	rwh	**Message Received Successfully** 0 No successful reception since last flag reset. 1 A message has been received successfully. RXOK must be reset by software (write 0). Writing 1 has no effect.

Field	Bits	Type	Description
ALERT	5	rwh	**Alert Warning** The ALERT bit is set upon the occurrence of one of the following events (the same events which also trigger an alert interrupt if NCR.ALIE is set): 1. A change of bit BOFF in the CAN Node Status Register. 2. A change of bit EWRN in the CAN Node Status Register. 3. A List Length Error, which also sets bit LLE in the CAN Node Status Register. 4. A List Object Error, which also sets bit LOE in the CAN Node Status Register. 5. Bit INIT has been set by the MultiCAN. ALERT must be reset by software (write 0). Writing 1 has no effect.
EWRN	6	rh	**Error Warning Status** 0 No warning limit exceeded. 1 One of the error counters NECNT.REC or NECNT.TEC reached the warning limit NECNT.EWRNLVL.
BOFF	7	rh	**Bus-off Status** 0 CAN controller is not in the bus-off state. 1 CAN controller is in the bus-off state.
LLE	8	rwh	**List Length Error** 0 No list length error since last flag reset. 1 A list length error has been detected during message acceptance filtering. The number of elements in the list that belongs to this CAN node differs from the list SIZE given in the list termination pointer. LLE must be reset by software (write 0). Writing 1 has no effect.

Field	Bits	Type	Description
LOE	9	rwh	**List Object Error** 0 No list object error since last flag reset. 1 A list object error has been detected during message acceptance filtering. A message object with wrong LIST index entry in the Message Object Control Register has been detected. LOE must be reset by software (write 0). Writing 1 has no effect.
SUSACK	10	rh	**Suspend Acknowledge** 0 The CAN node is not in suspend mode or a suspend request is pending, but the CAN node has not yet reached BUS IDLE or BUS OFF. 1 The CAN node is in suspend mode: The CAN node is inactive (bit NCR.INIT internally forced to 1) due to an OCDS suspend request.
0	[31:11]	r	**Reserved;** read as 0; should be written with 0.

Table 30-8 Encoding of the LEC Bit Field

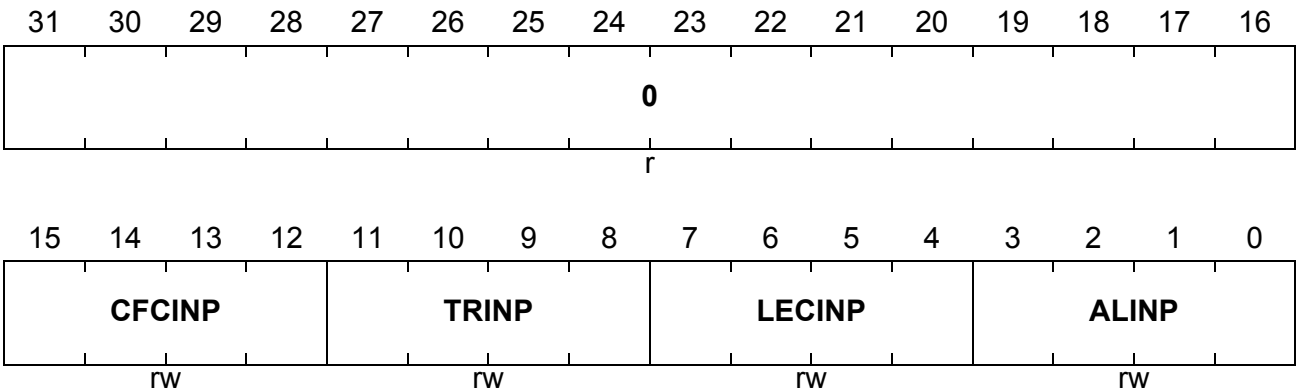
LEC Value	Signification
000_B	**No Error:** No error was detected for the last message on the CAN bus.
001_B	**Stuff Error:** More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
010_B	**Form Error:** A ‘fixed format part’ of a received frame has the wrong format.
011_B	**Ack Error:** The transmitted message was not acknowledged by another node.
100_B	**Bit1 Error:** During a message transmission the CAN node tried to send a **recessive** level (1) outside the arbitration field and the acknowledge slot, but the monitored bus value was **dominant**.

Table 30-8 Encoding of the LEC Bit Field (cont'd)

LEC Value	Signification
101_B	**Bit0 Error:** Two different conditions are signaled by this code: 1. During transmission of a message (or acknowledge bit, active error flag, overload flag) the CAN node tried to send a **dominant** level (0), but the monitored bus value was **recessive**. 2. During bus-off recovery this code is set each time a sequence of 11 **recessive** bits has been monitored. The CPU may use this code as indication that the bus is not continuously disturbed.
110_B	**CRC Error:** The CRC checksum of the received message was incorrect.
111_B	**CPU Write to LEC:** Whenever the CPU writes to LEC, it takes the value 111. The actually written LEC value is ignored.

The NIPR register connects each interrupt request source of the CAN node to one of the up to 16 available interrupt output lines.

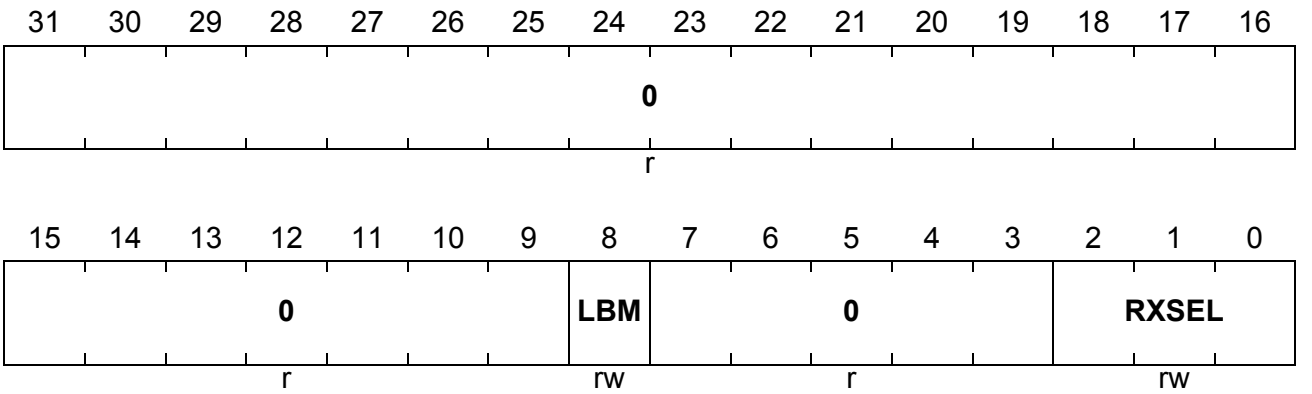
NIPRx (x = 0-3)
Node x Interrupt Pointer Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
ALINP	[3:0]	rw	**Alert Interrupt Enable** Number of interrupt output line INT_Om (m = 0-15) reporting the “Alert Interrupt Request”, if enabled by NCR.ALIE = 1.
LECINP	[7:4]	rw	**Last Error Code Interrupt Node Pointer** Number of interrupt output line INT_Om (m = 0-15) reporting the “Last Error Interrupt Request”, if enabled by NCR.LECIE = 1.
TRINP	[11:8]	rw	**Transfer OK Interrupt Node Pointer** Number of interrupt output line INT_Om (m = 0-15) reporting the “Transfer Interrupt Request”, if enabled by NCR.TRIE.
CFCINP	[15:12]	rw	**Frame Counter Interrupt Node Pointer** Number of interrupt output line INT_Om (m = 0-15) reporting the “Frame Counter Overflow Interrupt Request”, if enabled by NFCR.CFCIE = 1.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

The NPCR register configures the CAN bus transmit/receive ports. NPCRx may be written only if bit NCRx.CCE is set.

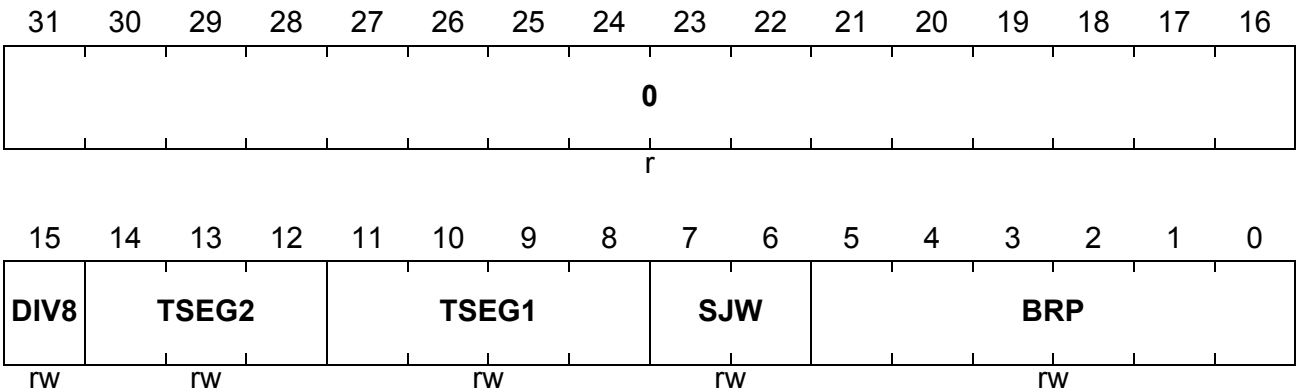
NPCRx (x = 0-3)
Node x Port Control Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RXSEL	[2:0]	rw	**Receive Select** RXSEL selects one out of 8 possible receive inputs. CAN traffic is performed through the selected input. The other inputs are ignored. See also **Section 30.3.5**.
LBM	8	rw	**Loop Back Mode** 0 Loop back mode is disabled. 1 Loop back mode is enabled. This node is connected to an internal (virtual) loop back CAN bus. All CAN nodes which are in loop back mode are connected to this virtual CAN bus so that they can communicate with each other internally. The external transmit line is forced **recessive** in loop back mode.
0	[7:3], [31:9]	r	**Reserved;** read as 0; should be written with 0.

The NBTR register contains all parameters to setup the bit timing for the CAN transfer. NBTRx may be written only if bit NCRx.CCE is set.

NBTRx (x = 0-3)
Node x Bit Timing Register **Reset Value: 0000 0000$_H$**

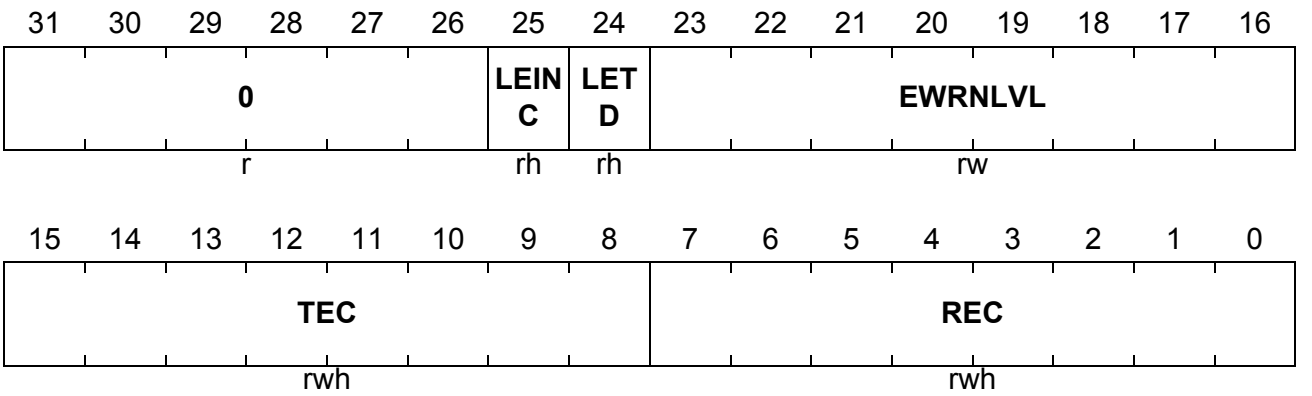


Field	Bits	Type	Description
BRP	[5:0]	rw	**Baud Rate Prescaler** The duration of one time quantum is given by (BRP + 1) clock cycles if DIV8 = 0. The duration of one time quantum is given by 8 × (BRP + 1) clock cycles if DIV8 = 1.
SJW	[7:6]	rw	**(Re)Synchronization Jump Width** (SJW + 1) time quanta are allowed for resynchronization.
TSEG1	[11:8]	rw	**Time Segment Before Sample Point** (TSEG1 + 1) time quanta is the user defined nominal time between the end of the synchronization segment and the sample point. It includes the propagation segment, which takes into account signal propagation delays. The time segment may be lengthened due to resynchronization. Valid values for TSEG1 are 2 to 15.
TSEG2	[14:12]	rw	**Time Segment After Sample Point** (TSEG2 + 1) time quanta is the user defined nominal time between the sample point and the start of the next synchronization segment. It may be shortened due to resynchronization. Valid values for TSEG2 are 1 to 7.

Field	Bits	Type	Description
DIV8	15	rw	**Divide Prescaler Clock by 8** 0 A time quantum lasts (BRP + 1) clock cycles. 1 A time quantum lasts 8 × (BRP + 1) clock cycles.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

The NECNT register contains the CAN receive and transmit error counter as well as some additional bits to ease error analysis. NECNTx may be written only if bit NCRx.CCE is set.

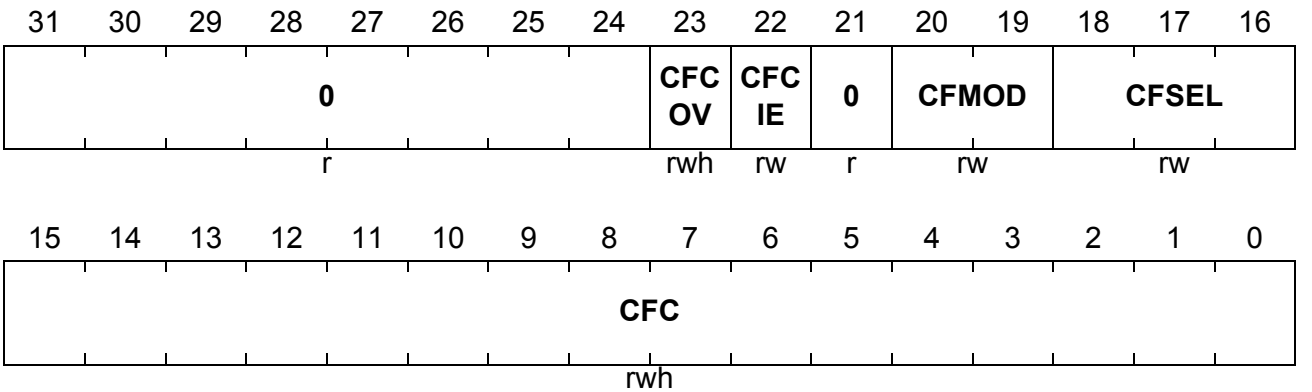
NECNTx (x = 0-3)
Node x Error Counter Register **Reset Value: 0060 0000$_H$**



Field	Bits	Type	Description
REC	[7:0]	rwh	**Receive Error Counter** Bit field REC contains the value of the receive error counter of the CAN node.
TEC	[15:8]	rwh	**Transmit Error Counter** Bit field TEC contains the value of the transmit error counter of the CAN node.
EWRNLVL	[23:16]	rw	**Error Warning Level** Bit field EWRNLVL defines the threshold value (warning level, default 96) to be reached in order to set the corresponding error warning bit NSR.EWRN.
LETD	24	rh	**Last Error Transfer Direction** 0 The last error occurred while the CAN node was receiver (REC has been incremented). 1 The last error occurred while the CAN node was transmitter (TEC has been incremented).
LEINC	25	rh	**Last Error Increment** 0 The last error led to an error counter increment of 1. 1 The last error led to an error counter increment of 8.
0	[31:26]	r	**Reserved;** read as 0; should be written with 0.

The NFCR register contains the actual value of the frame counter as well as control and status bits of the frame counter.

NFCRx (x = 0-3)
Node x Frame Counter Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CFC	[15:0]	rwh	**CAN Frame Counter** In Frame Count Mode this bit field contains the frame count value. In TimeStamp Mode this bit field contains the captured bit time count value, captured with the start of a new frame.

Field	Bits	Type	Description
CFSEL	[18:16]	rw	**CAN Frame Count Selection** This bit field selects the function of the frame counter for the chosen frame count mode. **Frame Count Mode** Bit 0 If Bit 0 of CFSEL is set then CFC is incremented each time a foreign frame (i.e. a frame not matching to a message object) has been received on the CAN bus. Bit 1 If Bit 1 of CFSEL is set then CFC is incremented each time a frame matching to a message object has been received on the CAN bus. Bit 2 If Bit 2 of CFSEL is set then CFC is incremented each time a frame has been transmitted successfully by the node. **Time Stamp Mode** The frame counter is incremented (internally) with the beginning of a new bit time. Its value is permanently sampled in the CFC field until the SOF bit of a new frame is detected. The sampled value is written to the corresponding message object. When the treatment of a message object is finished, the sampling continues. **Bit Timing Mode** The available bit timing measurement modes are shown in Table 30-9. If CFCIE is set then an interrupt on request node x (where x is the CAN node index) is generated with a CFC update.
CFMOD	[20:19]	rw	**CAN Frame Counter Mode** This bit field defines the operation mode of the frame counter. 00_B **Frame Count Mode:** The frame counter is incremented upon the reception and transmission of frames. 01_B **Time Stamp Mode:** The frame counter is used to count CAN bit times. 10_B **Bit Timing Mode:** The frame counter is used for analysis of the bit timing. 11_B Reserved

Field	Bits	Type	Description
CFCIE	22	rw	**CAN Frame Count Interrupt Enable** 0 CAN Frame Counter Overflow interrupt request is disabled. 1 CAN Frame Counter Overflow interrupt request is enabled.
CFCOV	23	rwh	**CAN Frame Counter Overflow Flag** Flag CFCOV is set upon a frame counter overflow (transition from $FFFF_H$ to 0000_H). In bit timing analysis mode CFCOV is set upon an update of CFC. An interrupt request is generated if CFCIE = 1. 0 No overflow has occurred since last flag reset. 1 An overflow has occurred since last flag reset. CFCOV must be reset by software.
0	[31:24]	r	**Reserved;** read as 0; should be written with 0.

Table 30-9 Bit Timing Analysis Modes (CFMOD = 10)

CFSEL	Measurement
000	Whenever a dominant edge (transition from 1 to 0) is monitored on the receive input the time (measured in clock cycles) between this edge and the most recent dominant edge is stored in CFC.
001	Whenever a recessive edge (transition from 0 to 1) is monitored on the receive input the time (measured in clock cycles) between this edge and the most recent dominant edge is stored in CFC.
010	Whenever a dominant edge is received as a result of a transmitted dominant edge the time (clock cycles) between both edges is stored in CFC.
011	Whenever a recessive edge is received as a result of a transmitted recessive edge the time (clock cycles) between both edges is stored in CFC.
100	Whenever a dominant edge that qualifies for synchronization is monitored on the receive input the time (measured in clock cycles) between this edge and the most recent sample point is stored in CFC.
101	With each sample point the time (measured in clock cycles) between this and the previous sample point is stored in CFC[11:0]. Additional information is written to CFC[15:12] with each sample point: CFC[15]: Transmit value of actual bit time CFC[14]: Receive sample value of actual bit time CFC[13:12]: CAN bus information (see **Table 30-10**)
111	Reserved

Table 30-10 CAN Bus State Information

CFC[13:12]	CAN Bus State
00	**NoBit** The CAN bus is idle, performs bit (de-) stuffing or is in one of the following frame segments: SOF, reserved bits, SRR, CRC, delimiters, first 6 EOF bits, IFS
01	**NewBit** This code represents the first bit of a new frame segment. The current bit is the first bit in one of the following frame segments: bit 10 (MSB) of standard ID (transmit only), RTR, IDE, DLC(MSB), bit 7 (MSB) in each data byte and the first bit of the ID extension.
10	**Bit** This code represents a bit inside a frame segment with a length of more than one bit (not the first bit of those frame segments which is indicated by NewBit). The current bit is processed within one of the following frame segments: ID bits (except first bit of standard ID for transmission and first bit of ID extension), DLC (3 LSB) and bits 6-0 in each data byte.
11	**Done** The current bit is in one of the following frame segments: Acknowledge slot, last bit of EOF, active/passive error frame, overload frame. Two or more directly consecutive Done codes signal an error frame.

30.2.3 Message Object Registers

The MOCTR register contains control bits for the CAN transfer and the message object link pointer. Each control bit has a corresponding bit in the CTRL field. A control bit is set by writing 1 to the corresponding bit in CTRL. It is cleared by writing 1 to the control bit directly. Any other combination leaves the control bit unchanged (see **Table 30-11**). After reset initialization the pointer PNEXT (read value of MOCTR[31:16]) points to message object n+1 (PNEXT = n+1), except for PNEXT of message object 127, which terminates the initial list (PNEXT = 127). Pointer PREV (read value of MOCTR[24:16]) initially points to message object n-1 (PPREV = n-1), except for PPREV of message object 0 which indicates the start of the initial list (PPREV = 0). This reset initialization means that all message objects initially belong to the list of unallocated elements.

MOCTRn (n = 0-127)
Message Object n Control Register **Reset Value: (n+1) << 24 + (n-1) << 16[1)]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTRL															
rwh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LIST				DIR	TXEN1	TXEN0	TXRQ	RXEN	RTSEL	MSGVAL	MSGLST	NEWDAT	RXUPD	TXPND	RXPND
rh				rw	rwh	rw	rwh	rw	rwh	rwh	rwh	rwh	rwh	rwh	rwh

1) Exceptions: Message Obj. 0: Reset val. of PPREV = 0, Message Obj. 127: Reset val. of PNEXT = 127.

Field	Bits	Type	Description
RXPND	0	rwh	**Receive Pending** RXPND set indicates that a CAN message has been received by the message object, either directly or via gateway copy action. RXPND is not cleared by the MultiCAN module but may be modified by the user.
TXPND	1	rwh	**Transmit Pending** TXPND set indicates that the message object has been transmitted successfully over the CAN bus. TXPND is not cleared by the MultiCAN module but may be modified by the user.

Field	Bits	Type	Description
RXUPD	2	rwh	**Receive Updating** RXUPD = 1 indicates that identifier, DLC and data of this message object are currently updated by the MultiCAN.
NEWDAT	3	rwh	**New Data** 0 No update of this message object since last flag reset. 1 The content of the message object has been updated. The MultiCAN module sets NEWDAT after storing a received CAN frame in this message object. NEWDAT is cleared by the MultiCAN when a CAN transmission of this message object has started. The user should set NEWDAT after storing new transmit data in the message object to prevent the automatic reset of TXRQ at the end of an ongoing transmission.
MSGLST	4	rwh	**Message Lost** MSGLST is set by the MultiCAN module whenever the MultiCAN module sets NEWDAT, but NEWDAT is already set (because it has not been reset by the user).
MSGVAL	5	rwh	**Message Valid** The message object is valid if and only if MSGVAL is set. Only valid message objects take part in CAN transfers.

Field	Bits	Type	Description
RTSEL	6	rwh	**Receive/Transmit Selected** **Frame Reception:** When this message object has been identified for storage of a CAN frame which is currently on reception, then RTSEL is set by the MultiCAN. It is verified that RTSEL is set before a received frame affects this message object. Thus the CPU may suppress a scheduled frame delivery to this object by clearing RTSEL. **Frame Transmission:** When this message object has been identified to be transmitted next, then RTSEL is set by the MultiCAN. It is verified that RTSEL is still set before the message object is actually set up for transmission and bit NEWDAT is cleared. The MultiCAN also verifies that RTSEL is still set before it modifies the message object due to the successful transmission of a frame. RTSEL only needs to be considered when the context of the message object is changed and interference with ongoing frame transfer shall be avoided. In all other cases RTSEL may be ignored. RTSEL has no impact on message acceptance filtering. RTSEL is not cleared by the MultiCAN.
RXEN	7	rwh	**Receive Enable** The message object may only receive a frame from the CAN bus if RXEN is set. The message object does not match any receiving frame if RXEN is not set. RXEN is only evaluated for receive acceptance filtering.
TXRQ	8	rwh	**Transmit Request** Request the transmission of the message object on the CAN bus. The transmit request is effective if TXRQ, TXEN0, TXEN1 and MSGVAL are set. This bit is automatically set if a matching remote frame has been received correctly. It is reset by hardware if the message object has been transmitted successfully and NEWDAT is not set again by software.

Field	Bits	Type	Description
TXEN0	9	rw	**Transmit Enable 0** The message object may only be transmitted if both TXEN0 and TXEN1 are set. The user may clear TXEN0 in order to inhibit the transmission of a message that is currently updated or to disable automatic response of remote frames.
TXEN1	10	rwh	**Transmit Enable 1** The message object may only be transmitted if both TXEN0 and TXEN1 are set. TXEN1 is used by the MultiCAN module for selecting the active message object in transmit FIFOs.
DIR	11	rw	**Message Direction** 0 Receive Object. With TXRQ = 1 a remote frame with the identifier of the message object is scheduled for transmission. On reception of a data frame with matching identifier the message is stored in this message object. 1 Transmit Object. If TXRQ = 1 this message object is scheduled for transmission as a data frame. On reception of a remote frame with matching identifier the TXRQ bit is set.
LIST	[15:12]	rh	**List Allocation** This field indicates the list to which the message object is allocated. LIST is updated by hardware when the list allocation of the object is modified by a panel command.

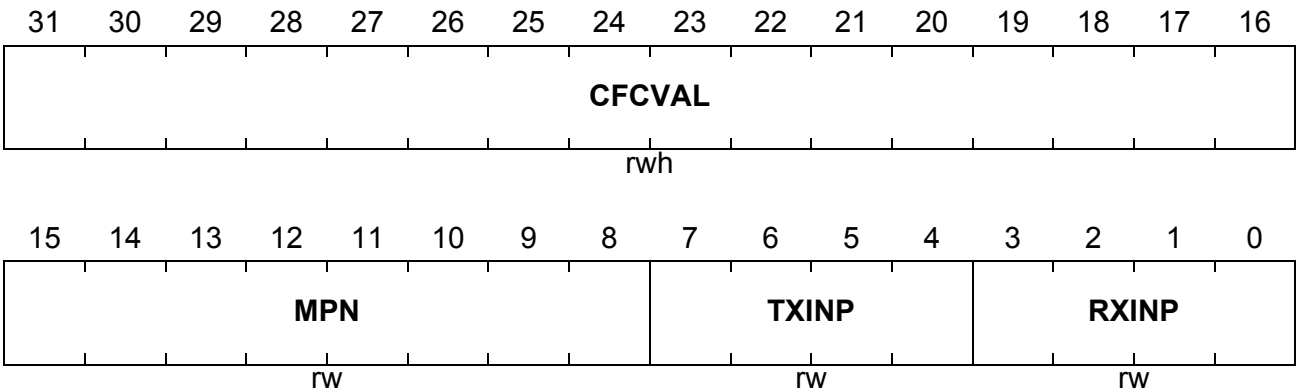
Field	Bits	Type	Description
CTRL	[31:16]	rwh	**Control Field** **Write Access:** Each control bit in the Message Object Control Register (bits 0-11 of MOCTR, e.g. bit 5: MSGVAL) is linked to a corresponding bit in the CTRL field. Control bit n (n in MOCTR[11:0]) is linked to bit n+16 (in MOCTR[27:16], i.e. in the CTRL field. Example: Bit MSGVAL is linked to MOCTR[21]). A control bit is set by writing value 1 into the corresponding bit in the CTRL field. It is cleared by means of writing value 1 directly to the control bit (Table 30-11). Bits [31:28] should be written zero (0). **Read Access:** MOCTR[23:16]: PPREV (Pointer to Previous object) MOCTR[31:24]: PNEXT (Pointer to Next object) PPREV holds the message number of the previous message object in the list structure. PNEXT holds the message number of the next message object in the list structure.

Table 30-11 Set/Reset of Message Control Bits

Value	Value of CTRL Bit	Function on Write
0 or no write access	**0** or no write access	Leave element unchanged
1	**0** or no write access	Reset element
0 or no write access	**1**	Set element
1	**1**	Leave element unchanged

The MOIPR register holds various pointer related to message interrupts as well as the frame counter value.

MOIPRn (n = 0-127)
Message Object n Interrupt Pointer Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RXINP	[3:0]	rw	**Receive Interrupt Node Pointer** Select the interrupt output line INT_Om (m = 0-15) for receive interrupts.
TXINP	[7:4]	rw	**Transmit Interrupt Node Pointer** Select the interrupt output line INT_Om (m = 0-15) for transmit interrupts.
MPN	[15:8]	rw	**Message Pending Number** This field selects the bit position of the bit in the message pending register to be set upon a receive/transmit interrupt.
CFCVAL	[31:16]	rwh	**CAN Frame Counter Value** When a message is stored in this message object or this message object has been successfully transmitted then NFCR.CFC is copied to CFCVAL.

The MOFCR register contains bits to select and to configure the function of the message object. It also holds the CAN data length code.

MOFCRn (n = 0-127)
Message Object n Function Control Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				DLC				STT	SDT	RMM	FRR EN	0	OVIE	TXIE	RXIE
r				rwh				rw	rw	rw	rw	r	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				DATC	DLCC	IDC	GDFS	0				MMC			
r				rw	rw	rw	rw	r				rw			

Field	Bits	Type	Description
MMC	[3:0]	rw	**Message Mode Control** Bit field MMC controls the functionality of the message object. 0000 Standard Message Object 0001 Receive FIFO Base Object 0010 Transmit FIFO Base Object 0011 Transmit FIFO Slave Object 0100 Gateway Source Object Else Reserved
GDFS	8	rw	**Gateway Data Frame Send** 1 TXRQ is set in the gateway destination object after the transfer of a data frame from the gateway source to the gateway destination. 0 TXRQ is not set in the destination object. Applicable only to Gateway Source Object.
IDC	9	rw	**Identifier Copy** IF IDC = 1 then the identifier of the gateway source object (after storing the received frame in the source) is copied to the gateway destination. Applicable only to Gateway Source Object.

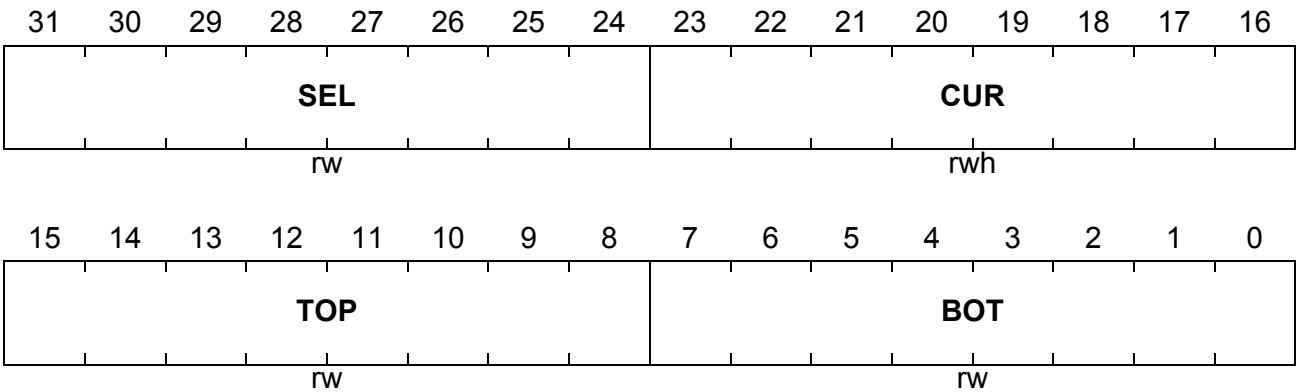
Field	Bits	Type	Description
DLCC	10	rw	**Data Length Code Copy** If DLCC = 1 then the data length code of the gateway source object (after storing the received frame in the source) is copied to the gateway destination. Applicable only to Gateway Source Object.
DATC	11	rw	**Data Copy** If DATC = 1 then the data field (registers MODATA0 and MODATA4) of the gateway source object (after storing the received frame in the source) is copied to the gateway destination. Applicable only to Gateway Source Object.
RXIE	16	rw	**Receive Interrupt Enable** If RXIE is set then a message interrupt request is generated with the reception of a CAN message, no matter whether the CAN message is received directly or indirectly via a gateway action. The interrupt is requested on interrupt node RXINP.
TXIE	17	rw	**Transmit Interrupt Enable** If TXIE is set then a message interrupt request is generated when this message object successfully transmitted a message over the CAN bus. The interrupt is requested on interrupt node TXINP.
OVIE	18	rw	**Overflow Interrupt Enable** IF OVIE = 1 then a FIFO full interrupt is generated when the pointer to the current object CUR reaches the value of SEL in the FIFO/Gateway Pointer Register. If this object is a receive FIFO base object then the FIFO full interrupt is requested on interrupt node TXINP. If this object is a transmit FIFO base object then the FIFO full interrupt is requested on interrupt node RXINP. For all other message object modes OVIE has no effect.

Field	Bits	Type	Description
FRREN	20	rw	**Foreign Remote Request Enable** Specifies if the TXRQ bit is set in this message object or in a foreign object referenced by the pointer CUR. 0 TXRQ of this message object is set upon the reception of a matching remote frame. 1 TXRQ of the message object referenced by the pointer CUR is set upon the reception of a matching remote frame.
RMM	21	rw	**Transmit Object Remote Monitoring** 0 Remote monitoring disabled: The identifier, IDE bit and DLC of the message object remain unchanged upon the reception of a matching remote frame. 1 Remote monitoring enabled: The identifier, DLC and IDE bit of a matching remote frame are copied to this transmit object in order to monitor incoming remote frames. Bit RMM only applies to transmit objects and has no impact on receive objects.
SDT	22	rw	**Single Data Transfer** If SDT = 1 and this object is not a FIFO base object then MSGVAL is reset when this object has taken part in a successful data transfer (receive or transmit). If SDT = 1 and this object is a FIFO base object then MSGVAL is reset when the pointer to the current object CUR reaches the value of SEL in the FIFO/Gateway Pointer Register. With SDT = 0, bit MSGVAL is not affected.
STT	23	rw	**Single Transmit Trial** If this bit is set then TXRQ is cleared upon transmission start of this message object. Thus no transmission retry is performed in case of transmission failure.
DLC	[27:24]	rwh	**Data Length Code** Valid values for the data length are 0 to 8. DLC > 8 leads to 8 data bytes, but the DLC code is not truncated upon reception or transmission of CAN frames.

Field	Bits	Type	Description
0	[7:4], [15:12], 19, [31:28]	r	**Reserved;** read as 0; should be written with 0.

The MOFGPR register contains a set of message object link pointer used for FIFO and gateway functionality.

MOFGPRn (n = 0-127)
Message Object n FIFO/Gateway Pointer Register **Reset Value: 0000 0000$_H$**

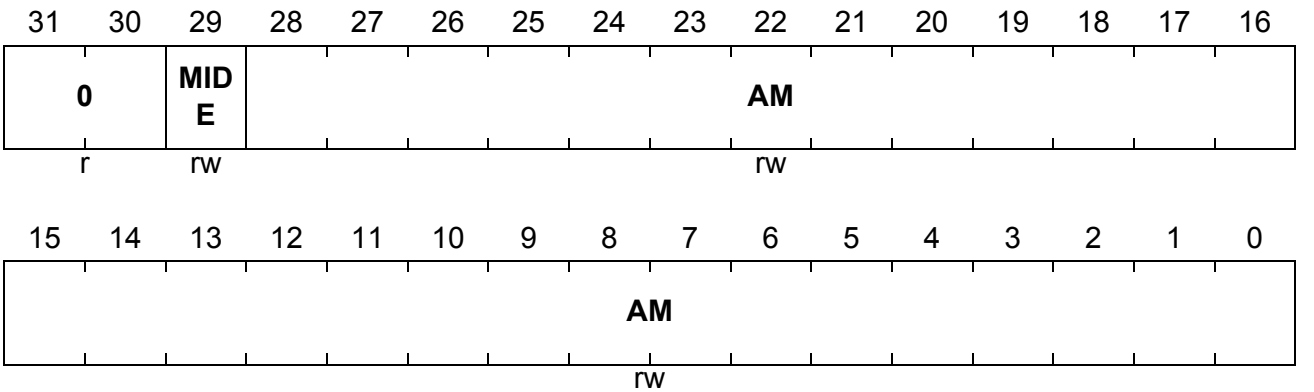


Field	Bits	Type	Description
BOT	[7:0]	rw	**Bottom Pointer** The Bottom Pointer points to the first element in a FIFO structure.
TOP	[15:8]	rw	**Top Pointer** The TOP pointer points to the last element in a FIFO structure.
CUR	[23:16]	rwh	**Current Object Pointer** The Current Object Pointer links to the actual target object within a FIFO/Gateway structure. After a FIFO/gateway operation, CUR is updated with the message number of the next message object in the list structure (given by PNEXT of MOCTR register) until it reaches the FIFO top element (given by TOP) when it is reset to the bottom element (given by BOT).
SEL	[31:24]	rw	**Object Select Pointer** The Object Select Pointer is the second (software) pointer to complement the hardware pointer CUR in the FIFO structure. SEL is used for monitoring purposes only.

Note: The pointers in this register must be set to objects assigned to the same CAN node. It is forbidden to refer to objects that are not in the linked list for the same CAN node.

Register MOAMR contains the mask bits for the acceptance filtering of the message object.

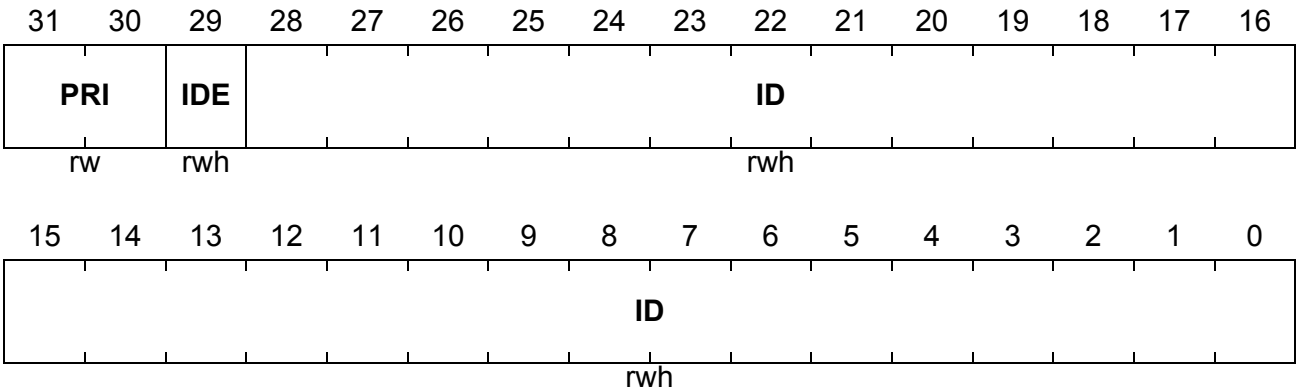
MOAMRn (n = 0-127)
Message Object n Acceptance Mask Register **Reset Value: 3FFF FFFF$_H$**



Field	Bits	Type	Description
AM	[28:0]	rw	**Acceptance Mask for Message Identifier** Mask to filter incoming messages with standard identifiers (AM[28:18]) or extended identifiers (AM[28:0]). For standard identifiers bits AM[17:0] are “don’t care”.
MIDE	29	rw	**Acceptance Mask Bit for Message IDE Bit** 0 This message objects accepts the reception of both standard and extended frames. 1 This message object only receives frames with matching IDE bit.
0	[31:30]	r	**Reserved;** read as 0; should be written with 0.

Register MOAR contains the CAN identifier of the message object.

MOARn (n = 0-127)
Message Object n Arbitration Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
ID	[28:0]	rwh	**CAN Identifier of Message Object** Identifier of a standard message (ID[28:18]) or an extended message (ID[28:0]). For standard identifiers bits ID[17:0] are “don’t care”.
IDE	29	rwh	**CAN IDE bit of Message Object** 0 Standard frame with 11-bit identifier 1 Extended frame with 29-bit identifier

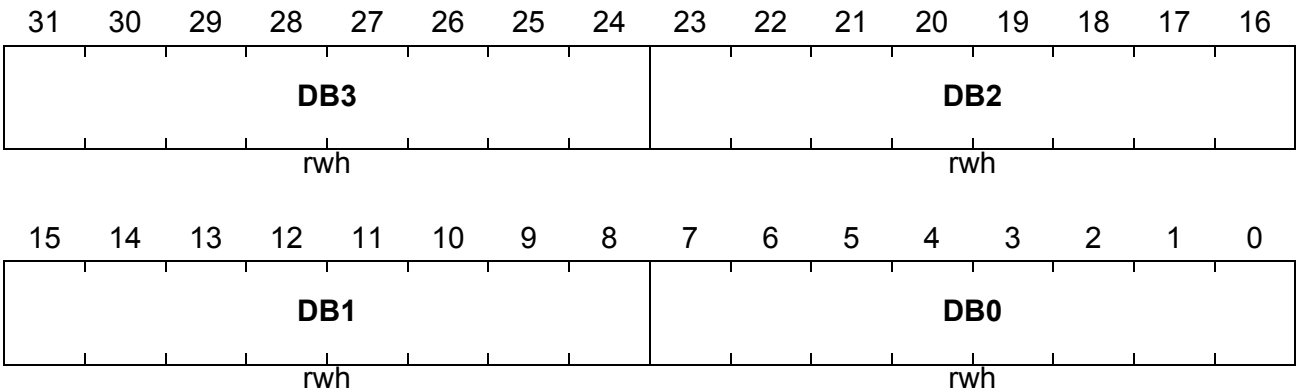
Field	Bits	Type	Description
PRI	[31:30]	rw	**Priority Class** PRI assigns one of the four priority classes 0, 1, 2, 3 to the message object, with lower PRI number meaning higher priority. Message objects with lower PRI value always win acceptance filtering for frame reception and transmission over message objects with higher PRI value. Acceptance filtering based on identifier/mask and list position is only performed between message objects of the same priority class. PRI also defines the acceptance filtering method for transmission: 00 Reserved 01 Transmit acceptance filtering is based on the list order, i.e. this message object is considered for transmission only if there is no other message object with valid transmit request (MSGVAL & TXEN0 & TXEN1 = 1) somewhere before this object in the list. 10 Transmit acceptance filtering is based on the CAN identifier, i.e. this message object is considered for transmission only if there is no other message object with higher priority identifier+IDE+DIR (with respect to CAN arbitration rules) somewhere in the list (see **Table 30-12**). 11 Transmit acceptance filtering is based on the list order (like PRI = 01).

Table 30-12 Transmit Priority based on CAN Arbitration Rules

Settings of Arbitrarily Chosen Message Objects A and B, where A has Higher Transmit Priority than B	Comment
A.MOAR[28:18] < B.MOAR[28:18] (11-bit standard identifier of A less than 11-bit standard identifier of B)	Messages with lower standard identifier have higher priority than messages with higher standard identifier. MOAR[28] is the most significant bit (MSB) of the standard identifier. MOAR[18] is the least significant bit of the standard identifier.
A.MOAR[28:18] = B.MOAR[28:18] A.MOAR.IDE = 0 (send standard frame) B.MOAR.IDE = 1 (send extended frame)	Standard frames have higher transmit priority than extended frames with equal standard identifier.
A.MOAR[28:18] = B.MOAR[28:18] A.MOAR.IDE = B.MOAR.IDE = 0 A.MOCTR.DIR = 1 (send data frame) B.MOCTR.DIR = 0 (send remote frame)	Standard data frames have higher transmit priority than standard remote frames with equal identifier.
A.MOAR[28:0] = B.MOAR[28:0] A.MOAR.IDE = B.MOAR.IDE = 1 A.MOCTR.DIR = 1 (send data frame) B.MOCTR.DIR = 0 (send remote frame)	Extended data frames have higher transmit priority than extended remote frames with equal identifier.
A.MOAR[28:0] < B.MOAR[28:0] A.MOAR.IDE = B.MOAR.IDE = 1 (29-bit identifier)	Extended frames have higher transmit priority than extended frames with lower identifier. MOAR[28] is the most significant bit (MSB) of the overall identifier (standard identifier MOAR[28:18] and identifier extension MOAR[17:0]). MOAR[0] is the least significant bit (LSB) of the overall identifier.

The MODATAL register contains the lowest four CAN data bytes. Unused data bytes are padded zero upon reception and ignored for transmission.

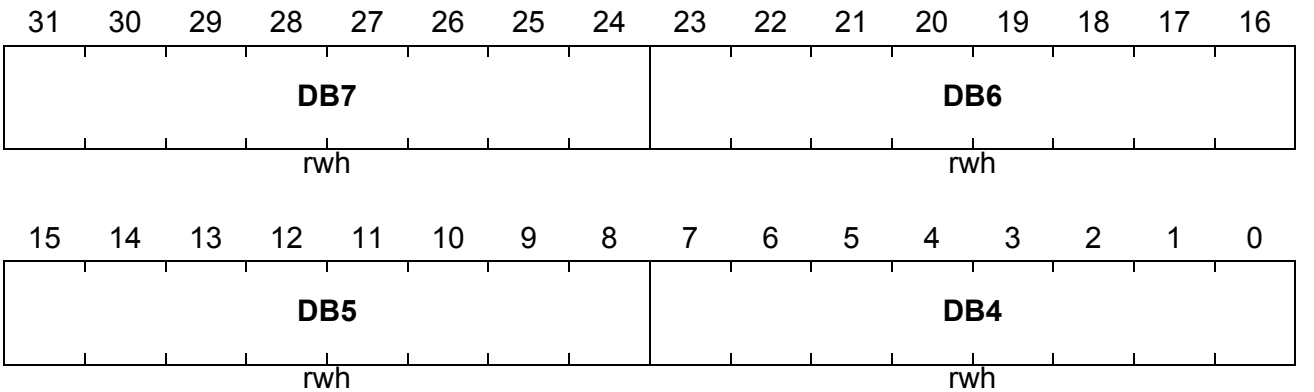
MODATALn (n = 0-127)
Message Object n Data Register Low **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DB0	[7:0]	rwh	**CAN Data Byte 0**
DB1	[15:8]	rwh	**CAN Data Byte 1**
DB2	[23:16]	rwh	**CAN Data Byte 2**
DB3	[31:24]	rwh	**CAN Data Byte 3**

The MODATAH register contains the highest four CAN data bytes. Unused data bytes are padded zero upon reception and ignored for transmission.

MODATAHn (n = 0-127)
Message Object n Data Register High **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DB4	[7:0]	rwh	**CAN Data Byte 4**
DB5	[15:8]	rwh	**CAN Data Byte 5**
DB6	[23:16]	rwh	**CAN Data Byte 6**
DB7	[31:24]	rwh	**CAN Data Byte 7**

30.2.4 Clock Control

The CAN clock frequency f_{CAN} of the functional blocks of the MultiCAN module is derived from the system clock f_{CLC} (= clock on the system bus). The fractional divider FDIV in the module is used to generate the CAN clock frequency for the bit timing calculation. The clock generation can be enabled/disabled by the fractional divider control bit field FDR.DM.

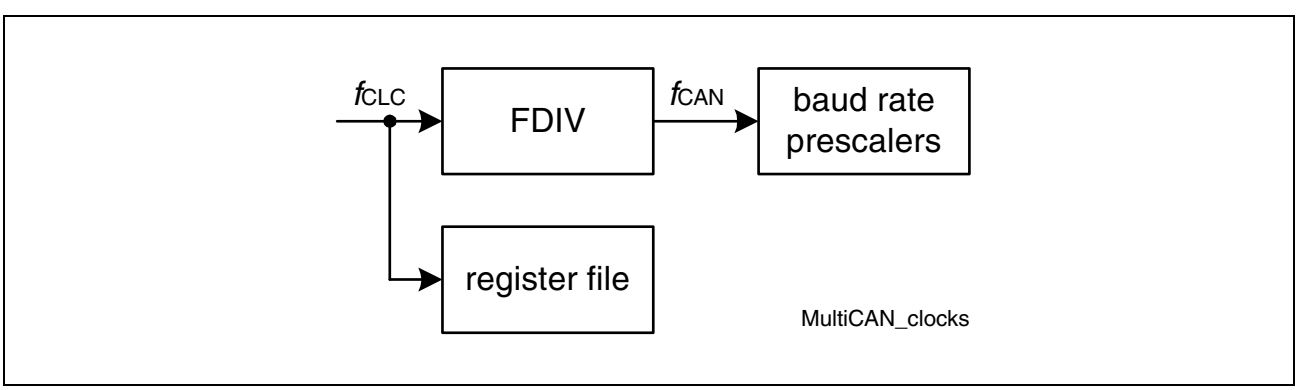


Figure 30-18 MultiCAN Clock Generation

The fractional divider FDIV output f_{CAN} is based on the system clock f_{CLC}, but only every n-th clock pulse is taken. The suspend signal (coming as acknowledge from the module as answer to the OCDS suspend request) freezes or resets the fractional divider.

30.2.5 Suspend Mode

The suspend mode can be triggered by the OCDS in order to freeze the state of the module and to have access to the registers (at least for read actions). There are several aspects related to the suspend mode:

- All actions are immediately stopped (“hard suspend”):
  The module clock is switched off as soon as the suspend line becomes active. This mode is supported by the fast switch off feature of the BPI. Write actions to the module are not supported and only combinatorial read actions deliver the desired data (the CAN RAM and the CAN registers cannot be accessed).
  In this mode, all further module actions are disabled and there is a very high probability that the communication with other devices is made impossible and that the CAN bus is blocked by the device in hard suspend mode (e.g. if the suspended CAN just sends a dominant level). A normal continuation when the suspend mode is left is not always possible and reset must be activated.
- The current action is finished (“soft suspend”):
  The module functions are stopped (clock is still running!) automatically after internal actions have been finished, for example after a CAN frame has been sent out. Due to this behavior, the communication network is not blocked due to the suspend mode of one communication partner. Furthermore, all registers are accessible for read and write actions. As a result, the debugger can stop the module actions and modify registers. These modifications are taken into account after the suspend mode is left. This mode is designed to be able to modify registers or to read them by the OCDS while the rest of the systems is still running and not corrupted by the suspend mode.

In the MultiCAN module a suspend mechanism is implemented allowing the individual freeze of CAN nodes. The fast switch off feature (hard suspend) of the BPI must not be activated by the user in order to support the soft suspend mode. In order to allow the required flexibility for the system, each CAN node can be individually enabled for the soft suspend mode.

The hard suspend feature can be enabled/disabled for the complete MultiCAN module, whereas the soft suspend feature can be enabled/disabled independently for each CAN node. The fractional divider disables the CAN clock only if all CAN nodes signal that they can be suspended. A CAN node that is not active can always be suspended.

30.2.6 Interrupt Structure

The general interrupt structure is shown in **Figure 30-19**. The interrupt event can trigger the interrupt generation. The interrupt pulse is generated independently from the interrupt flag in the interrupt status register. The interrupt flag can be reset by software by writing a 0 to it.

If enabled by the related interrupt enable bit in the interrupt enable register, an interrupt pulse can be generated at one of the 16 interrupt output lines INT_Ox of the module. If more than one interrupt source is connected to the same interrupt node pointer (in the interrupt node pointer register), the requests are combined to one common line.

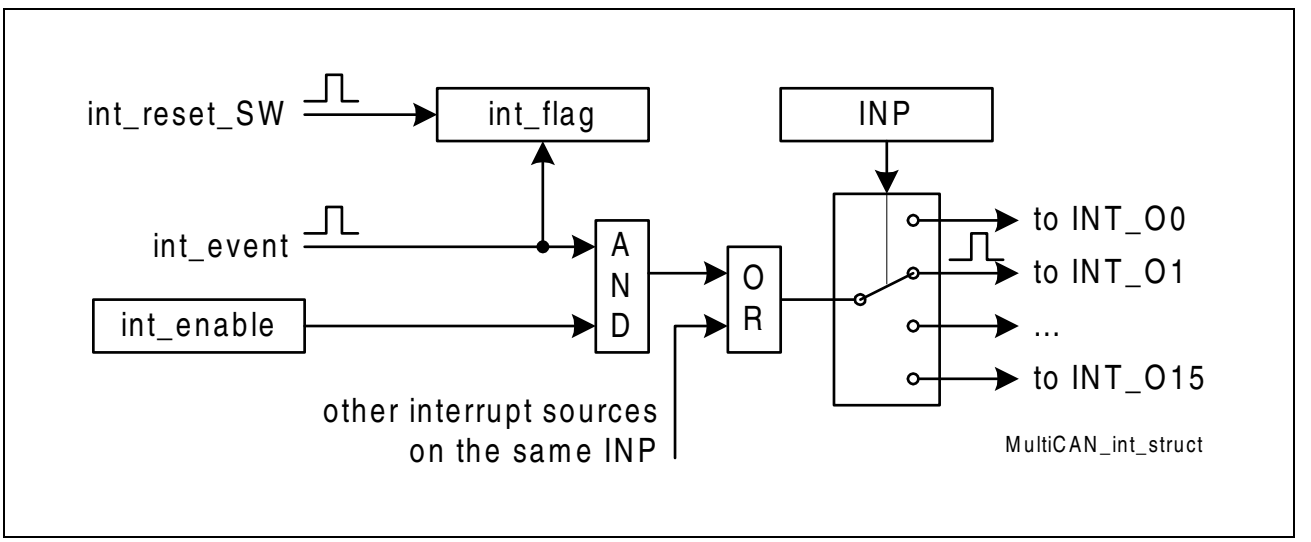


Figure 30-19 General Interrupt Structure

30.3 MultiCAN Module Implementation

This section describes CAN module interfaces with the clock control, port connections, interrupt control, and address decoding.

30.3.1 Interfaces of the CAN Module

Figure 30-20 shows the TC1130 specific implementation details and interconnections of the CAN module. The eight I/O lines of the CAN module kernel (two I/O lines of each CAN node) are connected to Port 0 and the four I/O lines of the Node 0 and Node 1 are connected to Port 1 also. The CAN module is further supplied by clock control, interrupt control, and address decoding logic.

DMA requests can be generated by the CAN module (INT_O0 to INT_O3).

The line INT_O15 is connected to the ECEN inputs of the fractional dividers of other modules in order to allow for synchronization.

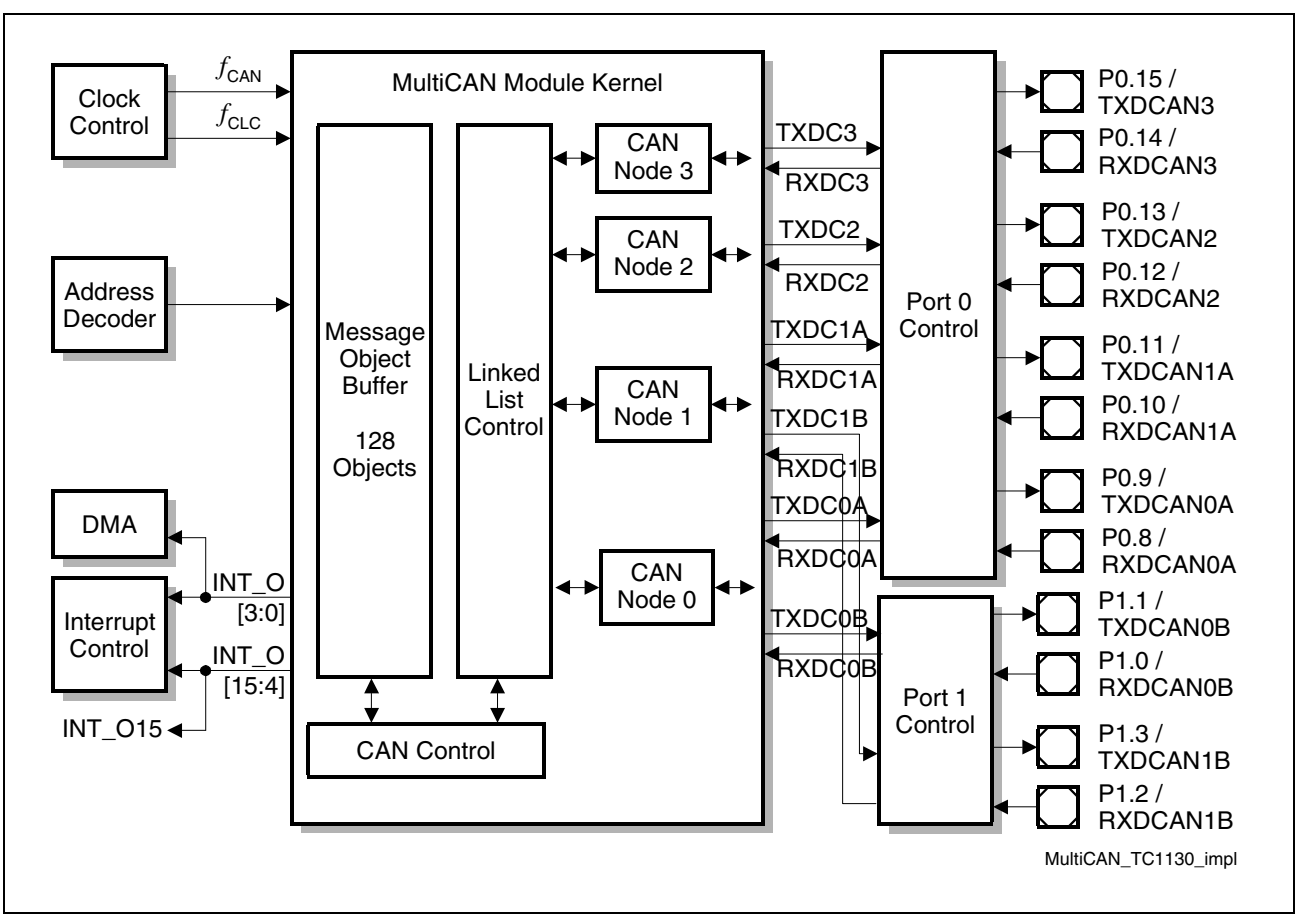


Figure 30-20 CAN Module Implementation and Interconnections

30.3.2 MultiCAN Module Related External Registers

Figure 30-21 summarizes the module related external registers which are required for MultiCAN programming (see also **Figure 30-17** for the module kernel specific registers).

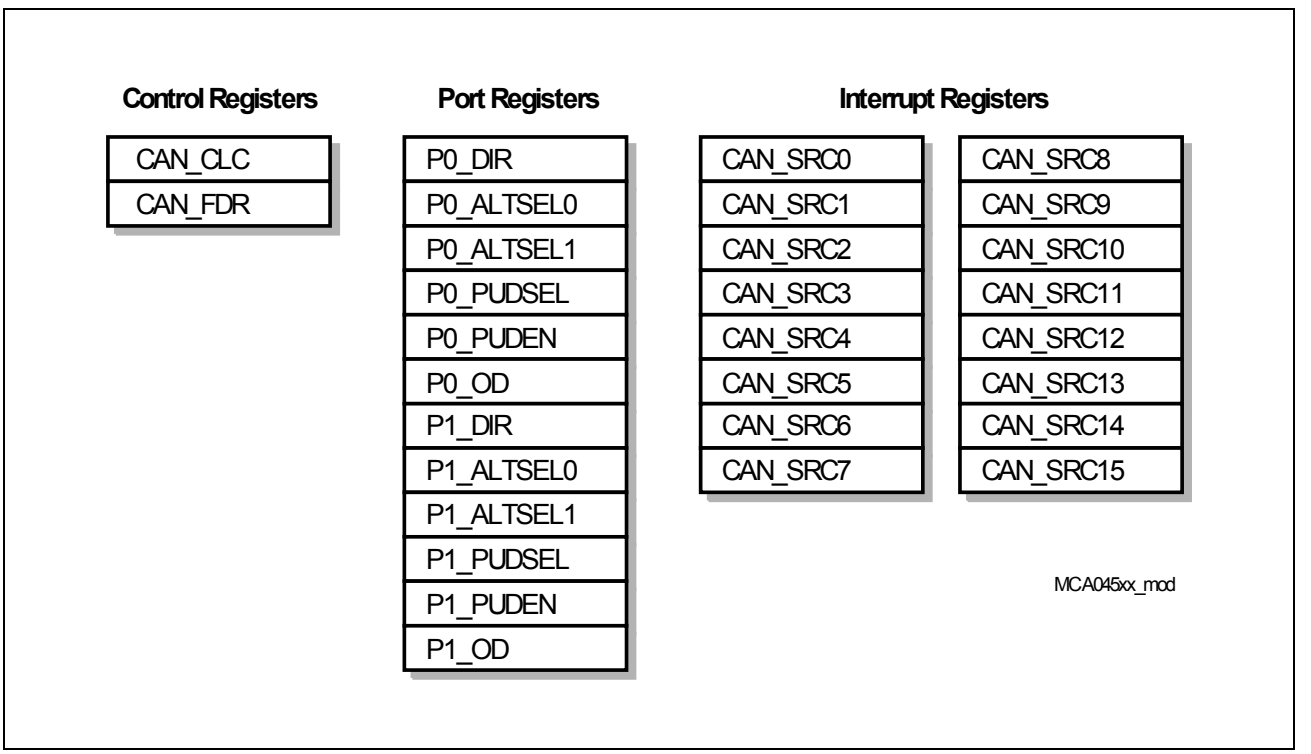


Figure 30-21 CAN Implementation Specific Special Function Registers

30.3.3 Module Clock Generation

As shown in **Figure 30-22**, the clock signals for the MultiCAN module are generated and controlled by a clock generation unit. This clock generation unit is responsible for the enable/disable control, the clock frequency adjustment, and the debug clock control. This unit includes two registers:

1. CAN_CLC: generation of the common control clock f_{CLC} for control tasks
2. CAN_FDR: Frequency control of the module timer clock f_{CAN}

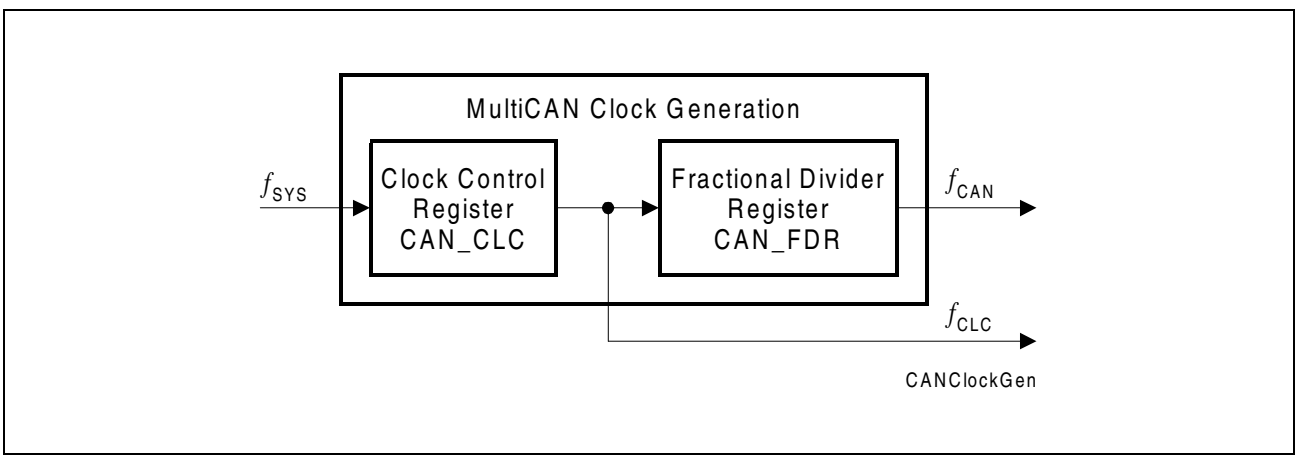


Figure 30-22 MultiCAN Module Clock Generation

The module control clock f_{CLC} is used inside the MultiCAN module kernel for control purposes such as e.g. for clocking of control logic and register operations. The frequency of f_{CLC} is identical to the system clock frequency f_{SYS}. The clock control register CAN_CLC allows f_{CLC} to be enabled/disabled under certain conditions.

The module timer clock f_{CAN} is used inside the MultiCAN module kernels as input clock for all timing relevant operations.

The frequency of f_{CAN} is defined by:

$$f_{CAN} = f_{SYS} \times \frac{1}{n} \text{ with n = 1024 - CAN\_FDR.STEP or}$$
$$f_{CAN} = f_{SYS} \times \frac{n}{1024} \text{ with n = 0 - 1023} \quad (30.1)$$

Note: The upper formula applies to normal divider mode of the fractional divider (CAN_FDR.DM = 01_B). The lower formula applies to fractional divider mode (CAN_FDR.DM = 10_B).

Note: Input signal ECEN of the MultiCAN fractional divider is wired to 0.

Table 30-13 Minimum Operating Frequencies [MHz]

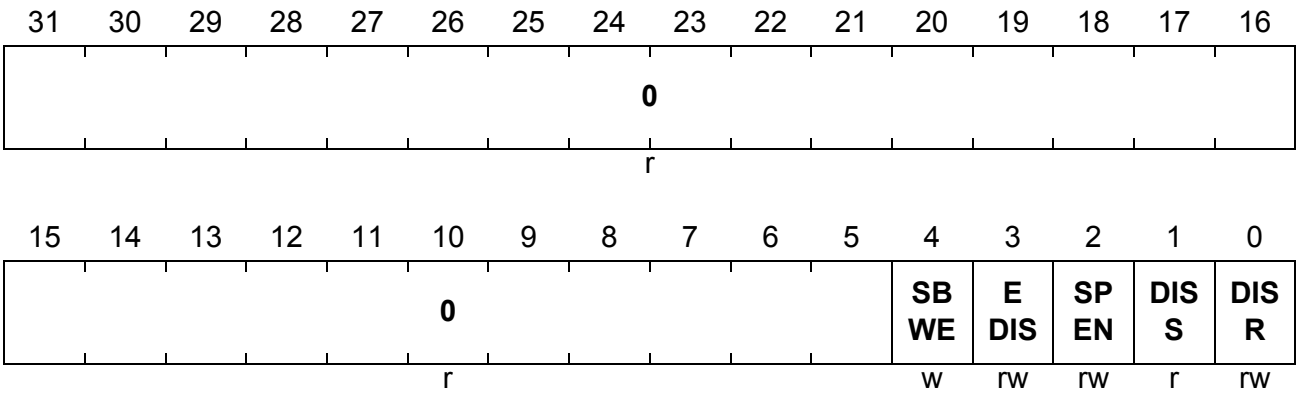
Number of Allocated Message Objects MO[1)]	With 1 CAN Node Active	With 2 CAN Nodes Active	With 3 CAN Nodes Active	With 4 CAN Nodes Active
16 MO	12	19	26	33
32 MO	15	23	30	37
64 MO	21	28	37	46
128 MO	40	45	50	55

1) Only those message objects have to be taken into account that are allocated to a CAN node. The unallocated message objects have no influence on the minimum operating frequency.

30.3.3.1 Clock Control Register

The clock control registers allow the control (enable/disable) of the module control clock f_{CLC}.

CAN_CLC
CAN Clock Control Register **Reset Value: 0000 0003$_H$**

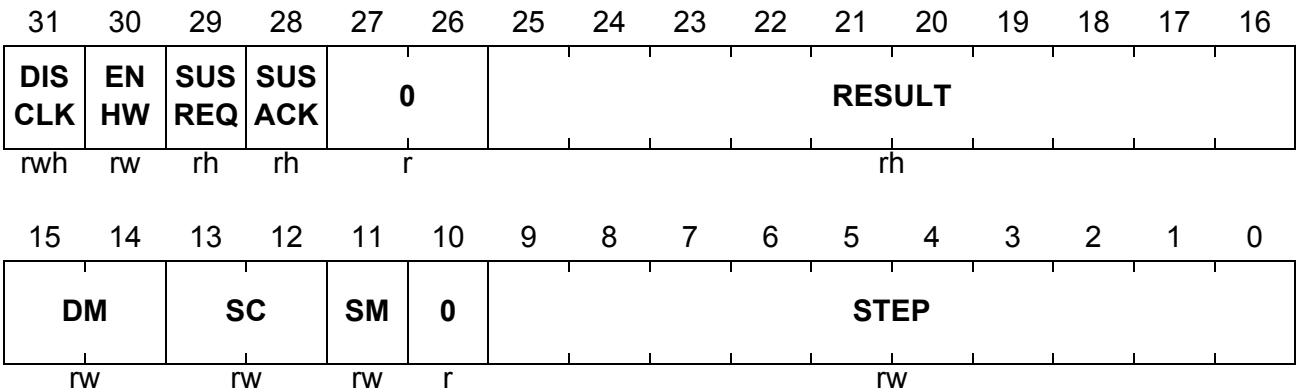


Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the module.
SPEN	2	rw	**Module Suspend Enable for OCDS** Used for enabling the suspend mode.
EDIS	3	rw	**External Request Disable** Used for controlling the external clock disable request.
SBWE	4	w	**Module Suspend Bit Write Enable for OCDS** Defines whether SPEN and FSOE are write protected.
0	[31:5]	r	**Reserved;** read as 0; should be written with 0.

30.3.3.2 Fractional Divider Register

The fractional divider register allows the programmer to control the clock rate and period of the module timer clock f_{CAN}.

CAN_FDR
CAN Fractional Divider Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
STEP	[9:0]	rw	**Step Value** Reload or addition value for RESULT.
SM	11	rw	**Suspend Mode** 0 Granted suspend mode 1 Immediate suspend mode
SC	[13:12]	rw	**Suspend Control** This bit field defines the behavior of the fractional divider in suspend mode.
DM	[15:14]	rw	**Divider Mode** This bit field selects normal divider mode, fractional divider mode, and off-state.
RESULT	[25:16]	rh	**Result Value** Bit field for the addition result.
SUSACK	28	rh	**Suspend Mode Acknowledge** Indicates state of SPNDACK signal.
SUSREQ	29	rh	**Suspend Mode Request** Indicates state of SPND signal.
ENHW	30	rw	**Enable Hardware Clock Control** Controls operation of ECEN input and DISCLK bit.
DISCLK	31	rwh	**Disable Clock** Hardware controlled disable for f_{OUT} signal.

Field	Bits	Type	Description
0	10, [27:26]	rw	**Reserved;** read as 0; should be written with 0.

30.3.4 Port Control

The interconnections between the MultiCAN module and the port I/O lines are controlled in the port logics. The following port control operations selections must be executed (additionally to the port input selection):

- Input/output function selection (DIR registers)
- Alternate function selection (ALTSEL0 and ALTSEL1 registers)
- Input/Output driver characteristic control (PUDSEL, PUDEN and OD registers)

Input/Output Function Selection

The port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as pull-up/down devices, port direction (input/output), open-drain, and alternate output selections. The I/O lines for the MultiCAN module are controlled by the port input/output control registers of Port0 and Port1. **Table 30-14** shows how bits and bit fields must be programmed for the required I/O functionality of the CAN I/O lines.

Table 30-14 CAN I/O Control Selection and Setup

Module	Port Lines	Input/Output Control Register Bits	I/O
CAN	P0.8/RXDCAN0A	P0_DIR.P8 = 0_B	Input
	P1.0/RXDCAN0B	P1_DIR.P0 = 0_B	Input
	P0.9/TXDCAN0A	P0_DIR.P9 = 1_B	Output
		P0_ALTSEL0.P9 = 1_B	
		P0_ALTSEL1.P9 = 0_B	
	P1.1/TXDCAN0B	P1_DIR.P1 = 1_B	Output
		P1_ALTSEL0.P1 = 0_B	
		P1_ALTSEL1.P1 = 1_B	
	P0.10/RXDCAN1A	P0_DIR.P10 = 0_B	Input
	P1.2/RXDCAN1B	P1_DIR.P2 = 0_B	Input
	P0.11/TXDCAN1A	P0_DIR.P11 = 1_B	Output
		P0_ALTSEL0.P11 = 1_B	
		P0_ALTSEL1.P11 = 0_B	
	P1.3/TXDCAN1B	P1_DIR.P3 = 1_B	Output
		P1_ALTSEL0.P3 = 0_B	
		P1_ALTSEL1.P3 = 1_B	
	P0.12/RXDCAN2	P0_DIR.P12 = 0_B	Input
	P0.13/TXDCAN2	P0_DIR.P13 = 1_B	Output
		P0_ALTSEL0.P13 = 1_B	
		P0_ALTSEL1.P13 = 0_B	
	P0.14/RXDCAN3	P0_DIR.P14 = 0_B	Input
	P0.15/TXDCAN3	P0_DIR.P15 = 1_B	Output
		P0_ALTSEL0.P15 = 1_B	
		P0_ALTSEL1.P15 = 0_B	

P0_DIR
Port 0 Direction Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 8-15)**	n	rw	**Port 0 Pin 8 - 15 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CAN I/O port control.

P1_DIR
Port 1 Direction Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-3)**	n	rw	**Port 0 Pin 0 - 3 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CAN I/O port control.

P0_ALTSELn (n = 0, 1)
Port 0 Alternate Select Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 30-15 Function of the Bits P0_ALTSEL0.Pn and P0_ALTSEL1.Pn (n = 9, 11, 13, 15)[1)]

P0_ALTSEL0.Pn	P0_ALTSEL1.Pn	Function
1	0	Alternate Select 1

1) Shaded bits and bit field are don't care for CAN I/O port control.

P1_ALTSELn (n = 0, 1)
Port 1 Alternate Select Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 30-16 Function of the Bits P1_ALTSEL0.Pn and P1_ALTSEL1.Pn (n = 1, 3)[1)]

P1_ALTSEL0.Pn	P1_ALTSEL1.Pn	Function
0	1	Alternate Select 2

1) Shaded bits and bit field are don't care for CAN I/O port control.

The MultiCAN ports also offer the possibility to configure the following output characteristics:

- Push/pull (optional pull-up/pull-down)
- Open drain with internal pull-up
- Open drain with external pull-up

P0_PUDSEL
Port 0 Pull-Up/Pull-Down Select Register **Reset Value: 0000 FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 8-15)**	n	rw	**Pull-Up/Pull-Down Select Port 0 Bit n**[1)] 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CAN I/O port control.

P1_PUDSEL
Port 1 Pull-Up/Pull-Down Select Register **Reset Value: 0000 FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-3)**	n	rw	**Pull-Up/Pull-Down Select Port 1 Bit n**[1)] 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CAN I/O port control.

P0_PUDEN
Port 0 Pull-Up/Pull-Down Enable Register

Reset Value: 0000 FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 8-15)**	n	rw	**Pull-Up/Pull-Down Enable at Port 0 Bit n**[1)] 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CAN I/O port control.

P1_PUDEN
Port 1 Pull-Up/Pull-Down Enable Register

Reset Value: 0000 FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-3)**	n	rw	**Pull-Up/Pull-Down Enable at Port 1 Bit n**[1)] 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CAN I/O port control.

P0_OD
Port 0 Open Drain Control Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	**P14**	**P13**	**P12**	**P11**	**P10**	**P9**	**P8**	**P7**	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 8-15)**	n	rw	**Port 0 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CAN I/O port control.

P1_OD
Port 1 Open Drain Control Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-3)**	n	rw	**Port 1 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for CAN I/O port control.

30.3.5 Connection of External Signals

The selection of the receive input pin is done independently for each CAN node by the bit field NPCRx.RXSEL, with x = 0-1.

Table 30-17 Receive Input Selection

Receive Input	Connected to	Selected by
RXDCAN0	P0.8	NPCR0.RXSEL = 000_B
	P1.0	NPCR0.RXSEL = 001_B
RXDCAN1	P0.10	NPCR1.RXSEL = 000_B
	P1.2	NPCR1.RXSEL = 001_B
RXDCAN2	P0.12	NPCR2.RXSEL = 000_B
RXDCAN3	P0.14	NPCR3.RXSEL = 000_B

The receive input lines RXDCANx (x = 1 - 0) are connected to:

- 1 (recessive) for NPCRx.RXSEL = 6 - 2
- 0 (dominant) for NPCRx.RXSEL = 7

The receive input lines RXDCANx (x = 3 - 2) are connected to:

- 1 (recessive) for NPCRx.RXSEL = 6 - 1
- 0 (dominant) for NPCRx.RXSEL = 7

30.3.6 Service Request Control Registers

Each of the sixteen interrupt output lines of the CAN module has its own service request control registers.

CAN_SRCm (m = 15-0)
CAN Service Request Control Register m **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETR	CLRR	SRR	SRE	0	TOS	0		SRPN							
w	w	rh	rw	r	rw	r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number**
TOS	10	rw	**Type of Service Control**
SRE	12	rw	**Service Request Enable**
SRR	13	rh	**Service Request Flag**
CLRR	14	w	**Request Clear Bit**
SETR	15	w	**Request Set Bit**
0	[9:8], 11, [31:16]	r	**Reserved;** read as 0; should be written with 0.

Note: Further details on interrupt handling and processing are described in chapter "Interrupt System" of the TC1130 System Units User's Manual.

30.3.7 DMA Requests

The DMA request lines of the CAN module are the interrupt output lines INT_O0 to INT_O3.

30.3.8 MultiCAN Module Register Address Map

In the TC1130, the registers of the MultiCAN module are located in the following address range:

- MultiCAN module: Module Base Address = F000 4000_H
  Module End Address = F000 $5FFF_H$
- Absolute Register Address = Module Base Address + Offset Address
  (offset addresses see **Table 30-5**)

Note: The complete and detailed address map of the MultiCAN modules is described in the chapter “Register Overview” of the TC1130 System Units User’s Manual.

31 Ethernet Controller

This chapter describes the Ethernet Controller of the TC1130 with the following sections:

- Functional description of the Ethernet Controller Kernel (see **Section 31.1**)
- Register descriptions of all Ethernet Controller Kernel-specific registers (see **Section 31.2**)
- TC1130 implementation-specific details and registers of the Ethernet Controller, including port connections and control, interrupt control and address decoding (see **Section 31.3**).

31.1 Ethernet Controller Kernel Description

31.1.1 Introduction

The Ethernet controller comprises the following functional blocks:

- Media Access Controller (MAC)
- Receive Buffer (RB)
- Transmit Buffer (TB)
- Data Management Unit in Receive Direction (DMUR)
- Data Management Unit in Transmit Direction (DMUT)

The MAC controller implements the IEEE 802.3 and operates at either 100 Mbit/s or 10 Mbit/s. RB and TB provide on-chip data buffering, whereas DMUR and DMUT perform data transfer from/to the shared memory.

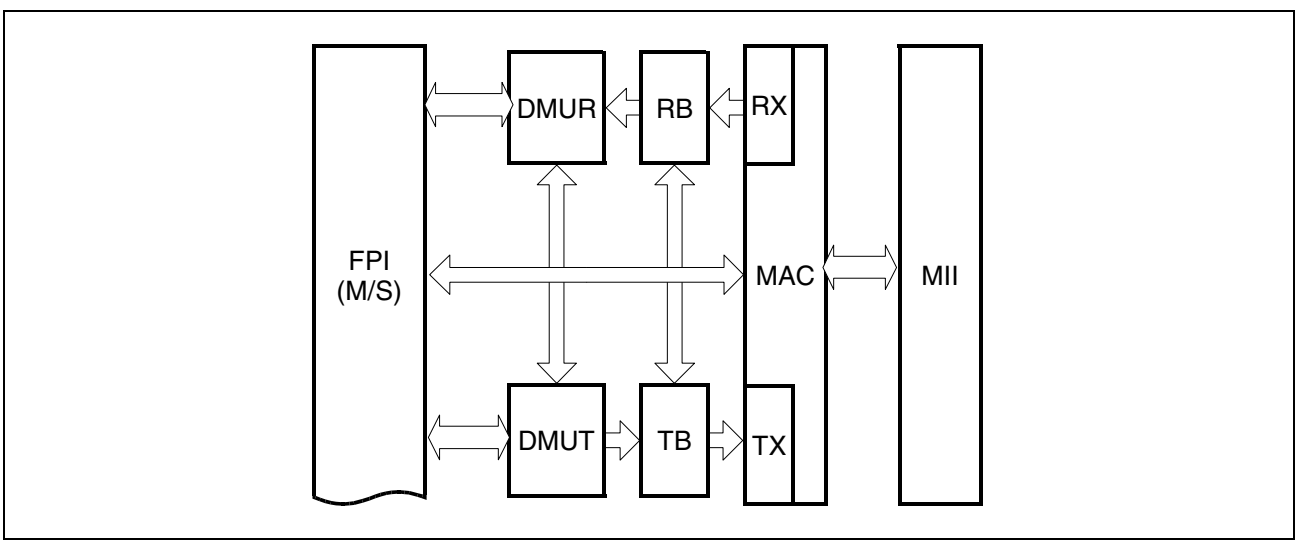


Figure 31-1 Ethernet Controller Functional Block

Two interfaces are provided by the device:

- MII interface for connection of Ethernet PHYs
- Master/slave Fast FPI Bus Interface for connection to the on-chip system bus for data transfer as well as configuration.

Features

- Media Independent Interface (MII) according to IEEE 802.3
- Supports 10 or 100 Mbit/s MII-based Physical devices
- Supports Full Duplex Ethernet
- Supports data transfer between Ethernet Controller and SDRAM via EBU
- 256-word Receive Buffer and 256-word Transmit Buffer
- Supports burst transfers up to 8 × 32 Bytes

Figure 31-2 shows the Ethernet Controller kernel with the module specific interface connections.

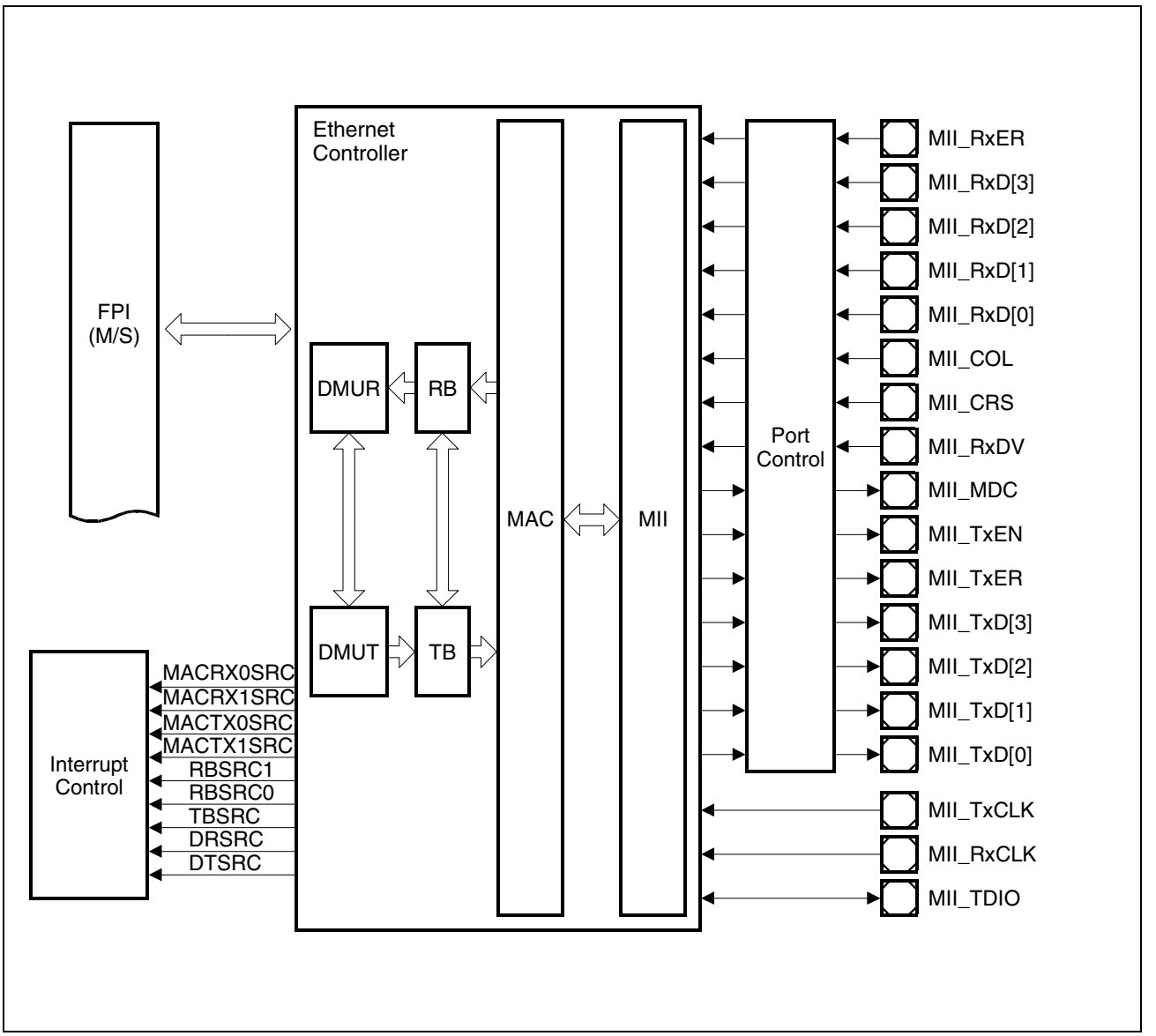


Figure 31-2 General Block Diagram of Ethernet Controller

31.1.2 Networking Ports

31.1.2.1 Media Independent Interface for Ethernet

An Ethernet Physical Layer entity (PHY) can be connected to the TC1130 via the Media Independent Interface (MII). The PHY performs all of the decoding on the incoming and outgoing data. The manner of decoding (Manchester for 10BASE-T, 4B/5B for 100BASE-X, or 8B/6T for 100BASE-T4) does not affect the MII. The MII expects raw data in the receive direction, starting with the preamble and ending with the CRC. The MII provides raw data for transmission, also starting with the preamble and ending with the CRC. The MAC Layer also generates collision data and transmits collision data (via MII to the PHY).

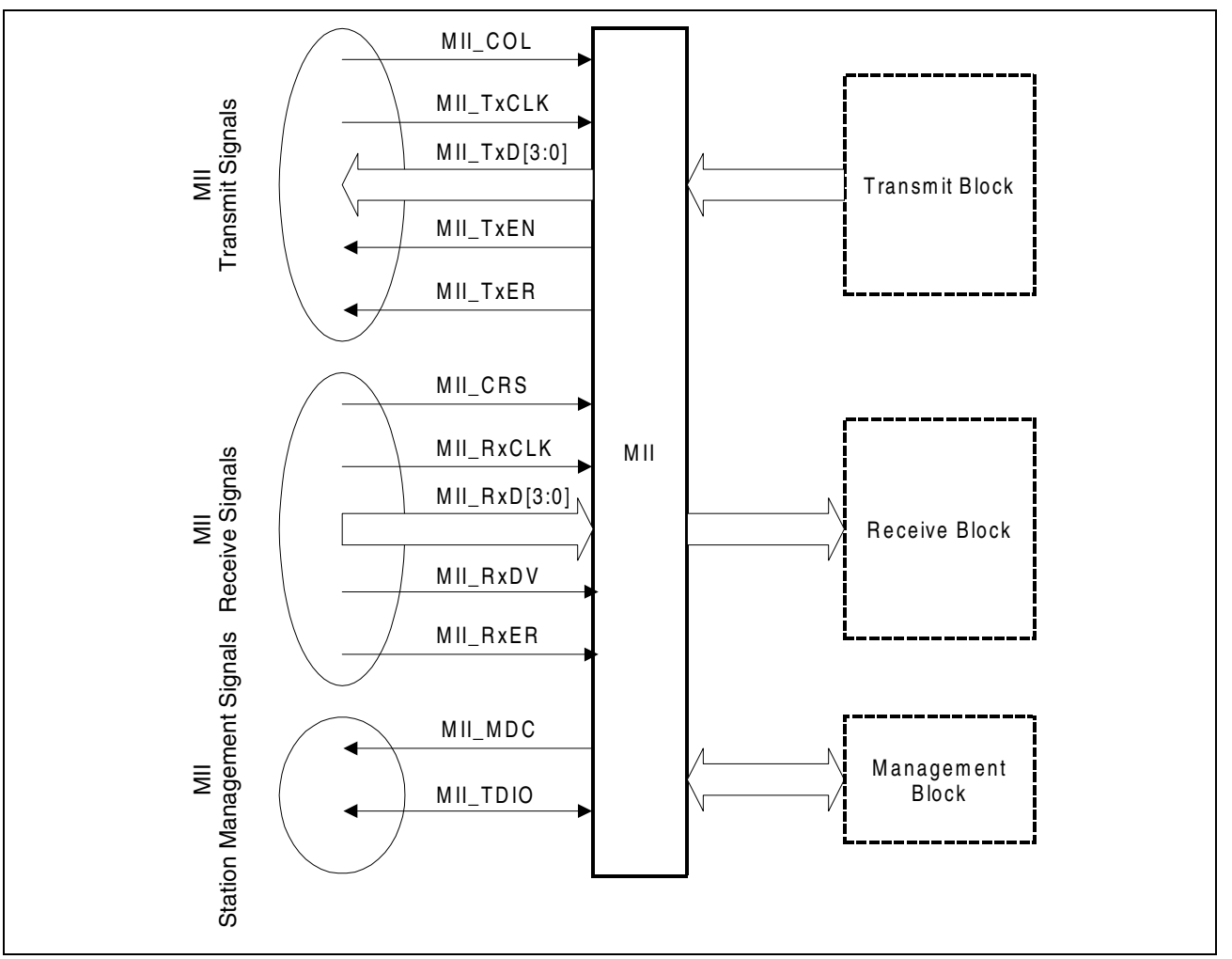


Figure 31-3 Ethernet Interface Signals

The MII operates at either 100 Mbit/s (@25 MHz) or 10 Mbit/s (@2.5 MHz). Data are aligned on nibble boundaries. For example, TXD[0] corresponds to the first bit to transmit on the physical medium and is the LSB of the first byte, followed by the fifth bit of that byte during the next clock. **Figure 31-3** provides an overview of the Ethernet Interface.

A serial driver is provided for reading and writing PHY registers for station management purposes. The data values transferred via the Station Management Interface are set or evaluated by the CPU via register access. Two registers (Management Block) are provided for Station Management:

- Station Management Data Register to write/read data to/from the attached PHY. Its format is defined by the MII section of the IEEE 802.3 Standard for 100-BASE-T, 100-Mbit/s Ethernet.
- Station Management Data Control and Address Register:
  - To select write/read command
  - To provide PHY device address
  - To provide PHY internal register address
  - To indicate status of command execution
  - To control transmission of the preamble

Setting options in station management registers does not affect the MAC controller.

The MII port supports a glue-less interface to industry standard Physical Media Dependent (PMD) devices.

This Interface has the following characteristics:

- Media independence
- Multi-vendor point of interoperability
- Support connection of MAC Layer and Physical (PHY) Layer devices
- Capable of supporting both 100 Mbit/s and 10 Mbit/s data rates
- Data and delimiter are synchronous to clock references
- Provides independent 4-bit wide transmit and receive data paths
- Supports connection of PHY Layer and Station Management (STA) devices
- Provides a simple management interface
- Capable of driving a limited length of shielded cable

31.1.2.2 Transmit MII Signals

Table 31-1 shows the MII signals by the Flow Control 100/10 Mbit/s Ethernet MAC for transmitting packets.

Table 31-1 Transmit MII Signals

Symbol	Name	Direction	Clock Domain
MII_COL	Collision Asserted asynchronously with minimum delay from the start of a collision on the medium	Input	–
MII_TXCLK	Transmit Clock MII_TXD[3:0] and MII_TXEN are driven off the rising edge of the MII_TXCLK by the CPU and sampled by the PHY on the rising edge of the MII_TXCLK.	Input	–
MII_TXD[3:0]	Transmit Data Transmit data is aligned on the nibble boundaries. MII_TXD[0] corresponds to the first bit to transmit on the physical medium and is the LSB of the first byte, followed by the fifth bit of that byte during the next clock.	Output	MII_TXCLK
MII_TXEN	Transmit Enable MII_TXEN provides precise framing for the data carried on MII_TXD[3:0]. It is active during the clock periods that MII_TXD[3:0] contains valid data to be transmitted, from preamble through CRC.	Output	MII_TXCLK
MII_TXER	Transmit Coding Error MII_TXER is driven synchronously to MII_TXCLK and is sampled continuously by the physical layer entity (PHY). If asserted for one or more MII_TXCLK periods, it causes the PHY to emit one or more symbols which are not part of the valid data or delimiter set somewhere in the frame being transmitted. Support of MII_TXER is required by PHY devices and optional for MAC devices. The MAC use MII_TXER to signal a transmit parity error, a late collision, or a FIFO underrun.	Output	MII_TXCLK

31.1.2.3 Receive MII Signals

Table 31-2 shows the MII signals supported by the Flow Control 100/10 Mbit/s Ethernet Stand-alone MAC for receiving packets. For a detailed description of these signals, see the MII sections of the IEEE 802 specification.

Table 31-2 Receive MII Signals

Symbol	Name	Direction	Clock Domain
MII_CRS	Carrier Sense Asserted asynchronously with minimum delay from the detection of a non-idle medium.	Input	–
MII_RXCLK	Receive Clock MII_RXCLK is a continuous clock. Its frequency is 25 MHz for 100 Mbit/s operation, and 2.5 MHz for 10 Mbit/s. MII_RXD[3:0], MII_RXDV and MII_RXER are driven by the PHY off the falling edge of the MII_RXCLK and sampled on the rising edge of the MII_RXCLK.	Input	–
MII_RXD[3:0]	Receive Data Receive data is aligned on the nibble boundaries. MII_RXD[0] corresponds to the first bit received on the physical medium which is the LSB of the byte in one clock period and the fifth bit of that byte during the next clock.	Input	MII_RXCLK
MII_RXDV	Receive Data Valid PHY asserts MII_RXDV synchronously and holds it active during the clock periods the MII_RXD[3:0] contains valid received data. PHY asserts MII_RXDV no later than the clock period when it places the first nibble of the start frame delimiter (SFD) on MII_RXD[3:0]. If PHY asserts MII_RXDV prior to the first nibble of the SFD, the MII_RXD[3:] carries valid preamble symbols.	Input	MII_RXCLK
MII_RXER	Receive Error PHY asserts MII_RXER synchronously whenever it detects a physical medium error, e.g., a coding violation. PHY asserts MII_RXER only when it asserts MII_RXDV.	Input	MII_RXCLK

31.1.2.4 MII Station Management Signals

Table 31-3 shows the MII station management signals. Use of these signals for configuring a PHY or negotiating a link protocol is optional. The MAC CSR Interface provides a serial driver for reading and writing PHY registers. None of the data values transferred via the MDIO Interface are interpreted by the MAC.

Table 31-3 MII Station Management Signals

Symbol	Name	Direction
MII_MDC	Management Data Clock Timing reference for transfer of information on the MDIO signal.	Output
MII_MDIO	Management Data Input When a read command is being executed, data which is clocked out of the PHY will be presented on the input line. Management Data Output When the CPU is clocking control or data onto the MDIO line, this signal will carry the information.	Input/ Output

31.1.3 Data Management Unit

The TriCore has a flexible DMA Controller to transfer data from either the internal Receive Buffer to the shared memory (receive direction) or from the shared memory to the internal Transmit Buffer (transmit direction). Each Ethernet data packet or part of a packet stored in the memory is referenced by a descriptor. The descriptors form two independent link lists, one for receive direction and one for transmit direction; thus connecting all receive or transmit packets together. Packet data as well as descriptors are located in system memory. Descriptors are stored independently from the data buffers, thus allowing full scatter/gather assembly and disassembly of data packets.

Both the Data Management Unit (DMU) and the system CPU operate on these data structures. This provides an optimized way to transfer data packets between the system processor and the Ethernet controller. The address generator of the DMA Controller supports full link list handling.

Descriptor Concept

A descriptor is used to build a linked list, where each member of the linked list points to a data section. A descriptor consists of four WORDS. The first three WORDS, containing link and packet information, are provided by the system CPU and the last WORD contains status information, which is written when the Ethernet Controller has finished operation on a descriptor.

The data section itself can be of any size up to the maximum size of 65535 bytes per descriptor and is defined in the first WORD of a descriptor. Each logical data packet can be split into one or multiple parts, where each part is referenced by one descriptor, and all parts are referenced by a linked list of descriptors. The descriptor containing the last part of a data packet is marked with a frame end bit. The descriptor following the marked descriptor therefore contains the beginning of the next data packet (**Figure 31-4**). The last descriptor in a linked list is marked with a hold indication.

For ease of programming, the transmit descriptor and the receive descriptor are structured the same way, thus allowing to link a receive descriptor directly into the linked list of the transmit queues with minimum descriptor processing.

Although the Data Management Unit is 32-bit oriented, it is possible to begin a transmit data section at an uneven address. The two Least Significant Bits of the transmit data pointer determine the beginning of the data section and the number of bytes in the first WORD of the data section, respectively. In the receive direction, the address of the data sections must be WORD aligned.

Figure 31-4 illustrates the descriptor structure.

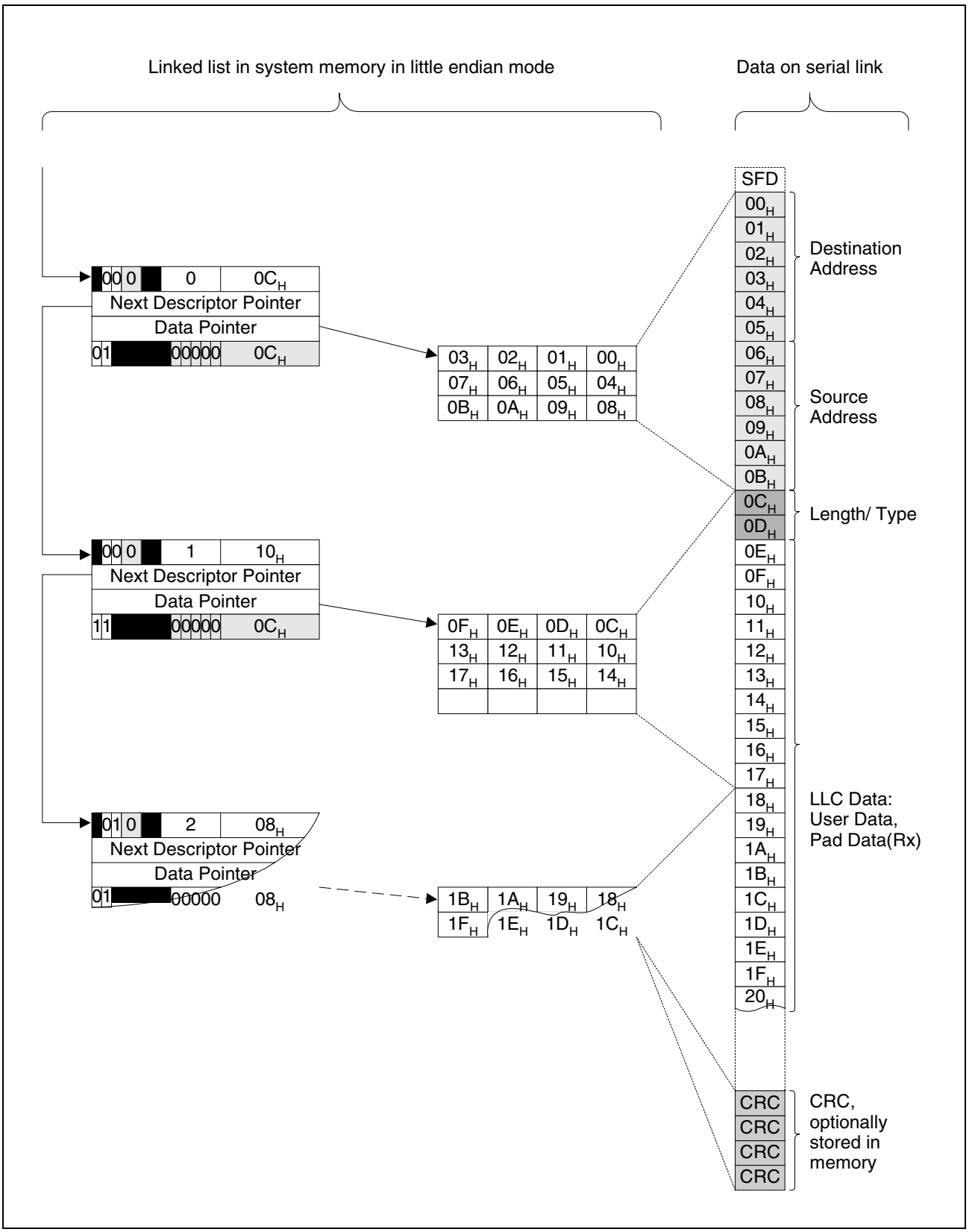


Figure 31-4 Descriptor Structure

31.1.3.1 Receive Descriptor

Each receive descriptor is initialized by the host CPU and is stored in system memory as part of a linked list. The TriCore chip reads a descriptor, when requested by the host via a receive command or after branching from one receive descriptor to the next receive descriptor. Each receive descriptor contains four WORDs, where the first three WORDs contain link and packet information and the last WORD contains status information. The status information will be written back to system memory by the TriCore chip as soon as a descriptor is processed. When the TriCore branches to a new descriptor, it reads the link and packet information entirely and stores the information in its on-chip channel database.

Receive Descriptor Structure

WORD ADDR.	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
00_H	0	HOLD	RHI	OFFSET(2:0)			0	0	0	0	DescriptorID(5:0)					
04_H	NextReceiveDescriptorPointer(31:2)															
08_H	ReceiveDataPointer(31:2)															
$0C_H$	FE	C	0	0	0	0	0	0	0	0	0	MFL	RFOD	CRC	ILEN	RAB

WORD ADDR.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00_H	NO(15:0)															
04_H	NextReceiveDescriptorPointer(31:2)														0	0
08_H	ReceiveDataPointer(31:2)														0	0
$0C_H$	BNO(15:0)															

HOLD: Hold Indication

HOLD indicates that a descriptor is the last element of a linked list containing valid information.

0: Next descriptor is available in the shared memory. After checking the HOLD bit, the DMU branches to the next receive descriptor.

1: This descriptor is the last one that is available for the current list. This means that the data section where this descriptor points to is the last data section which is available for data storage. After processing of the descriptor is finished, the DMU repolls the

descriptor one time to check if HOLD has already been cleared. If HOLD is still set, the corresponding receive channel is deactivated as long as the system CPU does not request a new activation via a ‘Receive Hold Reset’ command or forces the TriCore to branch to a new linked list via a ‘Receive Abort/Branch’ command.

Note: When repolling a descriptor, the TriCore checks the HOLD bit and the bit field NextReceiveDescriptorPointer. All other information is NOT updated in the internal channel database.

RHI: Receive Host Initiated Interrupt

This bit indicates that the TriCore will generate a ‘Receive Host Initiated’ interrupt vector after it has finished processing the descriptor.

0: Data Management Unit does not generate an interrupt vector after it has processed the receive descriptor.

1: Data Management Unit generates an interrupt vector as soon as all data bytes are transferred into the current data section and the status information is updated.

OFFSET: Offset Of Unused Data Section

This field is set by the CPU and is used by the Data Management Unit only at the beginning of a new frame. It allows the CPU to reserve memory space for an additional header. If the marked descriptor is the first one of a new packet, the DMU will write data at the address ReceiveDataPointer + Offset. Otherwise, this bit field is ignored.

Note: Offset x 4 must be smaller than NO.

DescriptorID

This bit field is read by the DMU and is written back in the corresponding interrupt status of a channel interrupt vector which is generated by the DMU. This value provides a link between the descriptor and the corresponding interrupt vector.

NO: Byte Number

This bit field defines the size of the receive data section allocated by the host. The maximum buffer length is 65535 bytes and it must be a multiple of 4 bytes. Data bytes are stored in the receive data section according to the selected mode (little endian or big endian).

Note: Please note that the device handles the status (CRC, frame status) of frame based protocols (Ethernet) internally in the same way as payload data. Therefore, byte number should include four bytes more than the maximum length of incoming frames. Nevertheless, the frame status will be deleted from the end of the data stream and be attached as a status word to the receive descriptor. The frame status will not be written to the data section.

NextReceiveDescriptorPointer

This pointer contains the start address of the next valid receive descriptor. After completion of the current receive descriptor, the DMU branches to the next receive descriptor to continue data reception.

System CPU can force the TriCore to branch to the beginning of a new linked list via the command ‘Receive Abort/Branch’. In this case, the receive descriptor address provided via register DRFRDA is used as the next receive descriptor pointer to be branched to.

ReceiveDataPointer

This pointer contains the start address of the receive data section. The start address must be WORD aligned.

FE: Frame End

It indicates that the current receive data section (addressed by ReceiveDataPointer) contains the end of a frame. This bit is set by the DMU after transferring the last data of a frame from the internal Receive Buffer into the receive data section which is located in the shared memory. Moreover, the bit field BNO and the status bits are updated, the complete (C) bit is set, and a ‘Frame End’ interrupt vector is generated.

C: Complete

This bit indicates that one of these conditions:

- Filling the data section has completed (with or without errors),
- Processing of this descriptor was aborted by a ‘Receive Abort/Branch’ command,
- The end of Ethernet frame was stored in the receive data section.

The complete bit releases the descriptor.

BNO: Byte Number of Received Data

The DMU writes the number of data bytes stored in the current data section into bit field BNO.

When the TriCore completes a data section which included the end of a frame (C bit and FE bit are set), or when the TriCore branches to a new linked list due to a ‘Receive Abort/Branch’ command, the status information bits RAB, ILEN, CRC, RFOD and MFL are updated. In the abort scenario, the C bit will always be set. Bit FE will be set if the end of an Ethernet frame has been received.

RAB: Receive Abort

This bit is set when one of these conditions is true:

- The incoming serial data stream contained an abort sequence
- Reception of Ethernet frames is halted by clearing RxEn bit or setting RxHalt bit

- An incoming frame was aborted by the command ‘Receive Abort/Branch’
- A channel is switched off while a frame is being received.

ILEN: Illegal Length

This bit is set, when one of these conditions is true:

- The length of the incoming data packet was not a multiple of eight bits, or
- The length of the received Ethernet frame in bits was not a multiple of eight bits and the CRC was invalid (AlignErr indication).

CRC: CRC Error

This bit is set, when one of these conditions is true:

- The checksum of an incoming data packet was different to the internally calculated checksum
- CRC at the end of an Ethernet packet did not match computed value, or the PHY asserted Rx_Er during Ethernet packet reception.

RFOD: Receive Frame Overflow

This bit is set when a receive buffer overflow occurred during data reception (Overflow Indication for Ethernet Frames).

MFL: Maximum Frame Length

This bit is set when the length of the incoming data packet exceeded the maximum allowed frame length (LongErr Indication for Ethernet Frames).

31.1.3.2 Data Management Unit Receive

Data Management Unit Receive transfers data from the internal Receive Buffer to the data sections in the shared memory. To fulfill the task, it must be initialized for operation. Relevant channel information for the Data Management Unit is the address pointer to the first receive descriptor, the configuration, and interrupt registers.

The first receive descriptor is fetched from system memory and is stored in the chip internal database when the Receive Buffer requests a data transfer for the first time. The descriptor contains a pointer to the data section, the size of the provided data section, and a pointer to the next receive descriptor.

The data transfer is requested as soon as a programmed Receive Buffer threshold is reached. This threshold is programmed during setup of the Receive Buffer. The task of the Data Management Unit is to calculate the maximum number of bytes that can be stored in the receive data section and to compare this with the length of the requested data transfer.

In the case that the requested transfer length from the receive buffer fits into the provided data section, the Data Management Unit transfers the data block to system memory in a sequence of 8 WORD bursts followed by 4-, 2-, 1-WORD burst(s), depending on the requested transfer length. If the requested transfer length exceeds the available space of the data section, the Data Management Unit branches to the data buffer of the next descriptor. Data packets are written to the data section until the given data section is filled or the end of a packet is reached.

If the data section in the shared memory is completely filled with data, the Data Management Unit updates the status word of the receive descriptor by setting the complete (C) bit and the number of bytes (BNO), stored in the data section. In this case, the number of bytes written to the data section equals the size of the data section.

If the data packet, which is written to system memory, contains the remaining part of a completely received packet, the Data Management Unit updates the status word of the receive descriptor by setting the complete bit together with the frame end (FE) bit. The BNO field is updated with the actual value of bytes written to the data section. If enabled, the Data Management Unit generates a ‘Frame End’ channel interrupt.

With the next receive buffer request, the Data Management Unit branches to the next receive descriptor, which was referenced in the next descriptor field of the current processed descriptor. To keep track of the linked list, the Data Management Unit provides the possibility to issue a ‘Receive Host Initiated’ interrupt, generated after the status word was updated. To enable this interrupt the bit RHI must be set in a descriptor.

Descriptor HOLD Operation

Processing of the descriptor list is controlled by the HOLD bit, which is located in the first WORD of each receive descriptor. The HOLD bit indicates that the marked descriptor is the last descriptor containing a valid data buffer. The data management unit will not branch to a next descriptor until the hold condition is removed or a ‘Receive Abort’ command forces the TriCore to branch to the beginning of a new linked list. Since the HOLD bit marks the last descriptor in a linked list, it may prevent that further received data packets can be written to system memory.

When a given data section is filled and it does not contain the end of a frame (frame based protocols) and the requested transfer length could not be satisfied, the data management unit polls the HOLD bit of the current receive descriptor once more. If the HOLD bit is removed, it branches to the next descriptor. When the HOLD bit is still 1, an internal poll bit is set and the data management unit does not branch to the next descriptor. Additionally, a ‘Hold Caused Receive Abort’ interrupt is generated. The status of the descriptor in the shared memory is aborted (RAB bit set) and the complete bit and the frame end bit are set in the receive descriptor. The rest of the frame will be discarded. If the HOLD bit remains set, further data of the same channel is discarded and a ‘Silent Discard’ interrupt for each discarded frame is generated with bits HRAB and RAB set.

If the current data section was filled and does contain the end of frame a ‘Frame End’ interrupt is generated and the descriptor is updated with the FE bit and the C bit. Therefore the status of this receive descriptor is error free. With the next request of the receive buffer, the Data Management Unit repolls the HOLD bit of the current receive descriptor. If the hold bit is removed, it branches to the next descriptor. If the HOLD bit is still 1, an internal poll bit is set. As long as the HOLD bit remains set, further data is discarded and for each discarded frame a ‘Silent Discard’ interrupt with bits HRAB and RAB set is generated.

When the Receive Buffer request matches exactly the remaining size of the data section and the data block does not contain the end of a packet, it is stored completely in the data section. The descriptor is updated immediately (C bit set). With the next receive buffer request, the Data Management Unit repolls the HOLD bit of the current receive descriptor. If the HOLD bit is removed, it branches to the next descriptor. If the HOLD Bit is still 1, an internal poll bit is set. Additionally, a ‘Hold Caused Receive Abort’ interrupt is generated and the rest of the frame is discarded. As long as the HOLD bit remains set, further data is discarded and for each discarded frame a ‘Silent Discard’ interrupt is generated.

The system CPU can remove the hold condition, when the next receive descriptor is available in shared memory. It is then recommended for CPU to execute a ‘Receive Hold Reset’ command, which will reactivate the channel. When the Receive Buffer requests a new data transfer, the data management unit will repoll the last receive descriptor. If the HOLD bit was removed, the Data Management Unit branches to the next receive descriptor pointed to by bit field NextReceiveDescriptor.

Note: Data from Receive Buffer is discarded until the end of a received frame is reached. As soon as the beginning of a new frame is received, the Data Management Unit starts to fill the data section.

If the CPU issues a ‘Receive Hold Reset’ command and does not remove the HOLD bit (erroneous programming), no action will take place.

31.1.3.3 Transmit Descriptor

The transmit descriptor in shared memory is initialized by the host CPU and is read afterwards by the TriCore. The address pointer to the first transmit descriptor is stored in the on-chip database, when requested to do so by the host CPU via the ‘Transmit Init’ command. The first three WORDs of a transmit descriptor are read when the Transmit Buffer requests a data transfer and then they are stored in the on-chip memory. Also, they are read when branching from one transmit descriptor to the next transmit descriptor. Therefore, all information in the next descriptor must be valid when the Data Management Unit branches to a descriptor. The last WORD of a transmit descriptor optionally is written by the TriCore when processing of a descriptor has finished.

Transmit Descriptor Structure

WORD ADDR.	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
00_H	FE	HOLD	THI	CEN	0	0	0	0	0	0	DescriptorID(5:0)					
04_H	NextTransmitDescriptorPointer(31:2)															
08_H	TransmitDataPointer(31:0)															
$0C_H$	0	C	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WORD ADDR.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00_H	NO(15:0)															
04_H	NextTransmitDescriptorPointer(31:2)														0	0
08_H	TransmitDataPointer(31:0)															
$0C_H$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FE: Frame End

It indicates that the current transmit data section (addressed by transmit data pointer) contains the end of a frame. After the last byte is read from system memory, this bit is passed to the Transmit Buffer and to the protocol machine. The bit FE informs the Transmit Buffer to move a stored frame to the protocol machine even if the programmed transmit forward threshold is not reached (see **Section 31.1.4.2**). The protocol machine is informed to append the checksum providing a transmit descriptor with FE = 0 and HOLD = 1 is an error.

HOLD Hold Indication

It indicates that this descriptor is the last valid element of a linked list.

0: Next descriptor is available in the shared memory. The DMU branches to the next descriptor as soon as processing of the current descriptor has finished.

1: The current descriptor is the last descriptor containing valid data in the data section. As soon as the DMU has transferred the data contained in the data section to the internal buffer, it tries one more time to read the descriptor. If the hold indication is still set, it stores further requests of the Receive Buffer in its database. The DMU can be reactivated by issuing a ‘Transmit Hold Reset’ command or by providing a new linked list via the ‘Transmit Abort/Branch’ command, in which case not served requests are processed.

Note: TriCore checks the HOLD bit and the NextTransmitDescriptorPointer bit field when repolling a descriptor. All other information is NOT updated in the internal channel database.

NO: Byte Number

The byte number defines the number of bytes stored in the data section to be transmitted. Thus, the maximum length of data buffer is 65535 bytes. In order to provide dummy transmit descriptors NO = 0 is allowed in conjunction with the FE bit set. In this case (NO = 0), a 'Transmit Host Initiated' interrupt and/or the C-bit will be generated/set when the DMU recognizes this condition. It is an error to set NO = 0 without FE bit set.

THI: Transmit Host Initiated Interrupt

This bit indicates that the TriCore will generate a 'Transmit Host Initiated' interrupt after it has finished operating on the descriptor.

0: DMU does not generate an interrupt after it has processed the transmit descriptor.

1: DMU generates an interrupt, as soon as all data bytes are transferred to the internal Transmit Buffer and the status information is updated.

DescriptorID

This bit field is read by the DMU and written back in the corresponding interrupt status which is generated by DMU. This value provides a link between the descriptor and the corresponding interrupt.

NextTransmitDescriptorPointer

This pointer contains the start address of the next transmit descriptor. It must be WORD aligned. After sending the indicated number of data bytes, the DMU branches to the next transmit descriptor. The transmit descriptor is read entirely at the beginning of transmission and stored in on-chip memory. Therefore, all information in the descriptor must be valid.

System CPU can force the TriCore to branch to the beginning of a new linked list via the command 'Transmit Abort/Branch'. In this case, the transmit descriptor address provided via register DTFTDA is used as the next transmit descriptor pointer to be branched to.

TransmitDataPointer

This 32-bit pointer contains the start address of the transmit data section. Although the DMU is WORD oriented, it is possible to begin transmit data section at byte addresses.

CEN: Complete Enable

This bit is set by the CPU if the complete bit mechanism is desired:

0: The DMU will NOT update the transmit descriptor with the C bit. In this mode, the use of the THI interrupt is recommended.

1: The DMU will set the C bit.

C: Complete

This bit is set by the DMU, when the bit CEN of a descriptor is set and when either:

- It completed reading a data section normally
- It was aborted by a ‘Transmit Off’ command or by a ‘Transmit Abort/Branch’ command.

The complete bit releases the descriptor.

31.1.3.4 Data Management Unit Transmit

Data Management Unit Transmit provides the interface between system memory on one hand and the internal Transmit Buffer on the other hand. The DMU handles requests of the Transmit Buffer, controls the address and burst length calculation, initiates data transfers from system memory to the Transmit Buffer, and handles the linked lists.

For initialization, the CPU programs the first transmit descriptor address, the interrupt and configuration registers, and starts the channel with the ‘Transmit Init’ command. The DMU then fetches the given information and stores it in its on-chip channel database.

The first transmit descriptor is fetched from system memory and is stored in the chip’s internal database the first time the Transmit Buffer requests data. It contains a pointer to the data buffer, the length of the data section, as well as a pointer to the next transmit descriptor. After the first descriptor is stored internally, a ‘Transmit Command Complete’ interrupt vector is generated.

Data transfers are requested as long as the number of empty locations is below a programmable refill threshold. The number of empty locations is reported from the Transmit Buffer to the DMU. The task of the DMU is to calculate the number of bytes that can be loaded from the data section based on the NO field of the transmit descriptor and to compare this with the number of bytes requested by the Transmit Buffer.

Depending on the bit field NO in the transmit descriptor, several read accesses must be performed by the DMU. The DMU stops serving the request as soon as the requested amount of data has been transferred to the Transmit Buffer, when a Frame End bit (FE) in the processed transmit descriptor is set, or when the channel is aborted using a ‘Transmit Abort’ command. This may result in open Transmit Buffer locations, but the DMU stores this information as open requests in the channel database and processes this requests continuously.

The DMU alternately serves requests issued by the Receive Buffer. If there are open requests for a channel, data transmission will be initiated. The procedure is the same as

described above. Data transmission stops if the requested amount of data has been served or when the FE bit field is set.

When the DMU completes reading a data section associated with a transmit descriptor, it updates the complete (C) bit in the status word of the transmit descriptor if the complete enable (CEN) bit is set. Additionally, a ‘Transmit Host Initiated’ interrupt is generated if the THI bit is set in the transmit descriptor. Afterwards, the DMU of the TriCore branches to the next transmit descriptor.

Descriptor HOLD Operation

The data transfer is controlled by the HOLD bit, located in the first WORD of a transmit descriptor. The HOLD bit indicates that the marked descriptor is the last descriptor in a linked list. The DMU will not branch to the next descriptor until the hold condition is removed or a ‘Transmit Abort’ command forces the TriCore to branch to a new linked list.

If the HOLD bit and the Frame End bit are set together in a descriptor, the DMU transfers all of the data from that data section to the Transmit Buffer and optionally sets the C-bit in the current transmit descriptor. When a new data transfer is requested (either from the Transmit Buffer or by an open request) the DMU repolls the descriptor. If the HOLD bit is removed, it will branch to the next transmit descriptor. If the HOLD bit is still set, the DMU is suspended for further operation. Subsequent requests from the Transmit Buffer will not be served, but the number of requested data is stored in the open request registers.

If the HOLD bit is detected in a descriptor and the Frame End bit is not set, the DMU will transfer all of the data of that data section to the Transmit Buffer. Afterwards, it generates a ‘Hold Caused Transmit Abort’ interrupt in order to inform the host CPU about the erroneous descriptor structure. In Ethernet Mode, the abort status is propagated to the Transmit Buffer and to the protocol machine, so that an abort sequence is sent on the serial side. Then, the DMU reads the transmit descriptor once more. If the HOLD bit is removed, it branches to the next transmit descriptor and proceeds with normal operation. Otherwise, when the HOLD bit is still set, the DMU is suspended for further operation and an internal poll bit is set. Subsequent requests from the Transmit Buffer will not be served, but the number of requested data is stored in the open request register.

The host CPU can remove the hold condition when the next transmit descriptor is available in system memory. Therefore, the CPU must execute a ‘Transmit Hold Reset’ command that will reactive the DMU. When the Transmit Buffer requests a new data transfer or when open requests are stored in the on-chip database, the DMU repolls the transmit descriptor and checks the HOLD bit again. If the HOLD bit is removed, it branches to next transmit descriptor.

If the CPU issues a ‘Transmit Hold Reset’ command and does not remove the HOLD bit (erroneous programming), no action will take place. Nevertheless, the CPU must always issue a ‘Transmit Hold Reset’ command when it removes the HOLD bit in a descriptor, whether or not the DMU has already seen the HOLD bit.

31.1.3.5 Byte Swapping

The TriCore operates by default as a little endian device. To support integration into big endian environments, the DMU provides an internal byte swapping mechanism that can be enabled via bits DRCONF.LBE and DTCONF.LBE. Big endian byte swapping applies only to the data section pointed to by the receive and transmit descriptors in shared memory.

Note: Byte swapping only affects the organization of packet data in system memory. All internal registers, as well as the descriptors, address pointers, or interrupt vectors are handled with little endian byte ordering.

Table 31-4 Example for Little/Big Endian with BNO = 3

BNO	Little Endian				Big Endian			
3	–	Byte 2	Byte 1	Byte 0	Byte 0	Byte 1	Byte 2	–

Table 31-5 Example for Little/Big Endian with BNO = 7

BNO	Little Endian				Big Endian			
7	Byte 3	Byte 2	Byte 1	Byte 0	Byte 0	Byte 1	Byte 2	Byte3
	–	Byte 6	Byte 5	Byte 4	Byte 4	Byte 5	Byte 6	–

Transmission Bit/Byte Ordering

Data is transmitted in increasing order beginning with Byte 0. Data received is stored starting with Byte 0. The position of Byte 0 depends on the selected endian mode.

Each byte consists of eight bits starting with Bit 0 (LSB) up to Bit 7 (MSB). Data on the serial line is transmitted starting with the LSB. The first bit received is stored in Bit 0.

31.1.3.6 Interrupt Queues

The DMU provides two separate interrupt status FIFOs, one for transmit direction and one for receive direction. Each FIFO can store up to 16 interrupt vectors to overcome potential bus latency.

31.1.4 Buffer Management

31.1.4.1 Internal Receive Buffer

The internal Receive Buffer provides buffering of frame data and status between the MAC Controller and the receive DMUs. Internal buffers are essential to avoid data loss due to the bus latency, especially in the presence of multiple devices on the same system bus, and to enable a minimized bus utilization through burst accesses.

The incoming data from the MAC Controller is stored in a receive central buffer. The buffer is written by the protocol handler every time a complete WORD is ready or the last byte of a frame has been received. The Transmit Buffer has an individual programmable threshold code that determines the number of WORDs after which a data transfer into the shared memory is generated. The threshold therefore defines the maximum burst length for a particular channel in receive direction. A data transfer is also requested as soon as a frame end has been reached. Programming the burst length to be greater than 1 WORD avoids accessing the FPI bus to frequently, thereby optimizing the use of this resource.

For real time applications with lowest possible latency (example: constant bit rate), a value of one WORD can be selected for the burst length.

The total size of the internal Receive Buffer is 1 Kbyte.

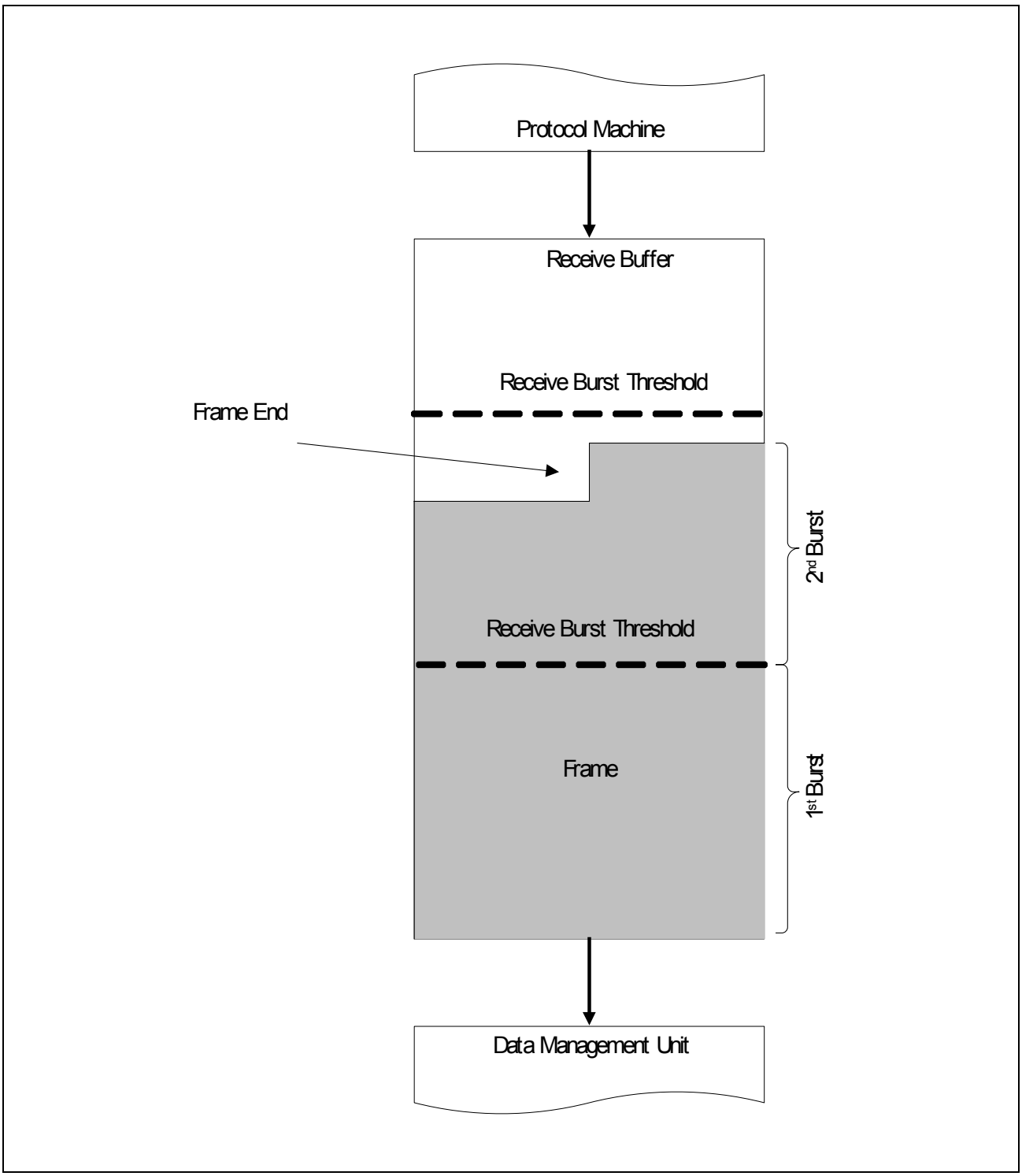


Figure 31-5 Receive Buffer Thresholds

For performance monitoring, the Receive Buffer is capable of monitoring utilization of the Receive Buffer and generating interrupts when certain fill thresholds have been reached.

31.1.4.2 Internal Transmit Buffer

The internal Transmit Buffer with a total size of 1 Kbyte stores protocol data before it is processed by the protocol machine. The Transmit Buffer is essential to ensure that enough data is available during transmission because FPI latency limits access to system memory. A programmable Transmit Buffer size and two programmable thresholds are configurable by the host CPU.

Note: The sum of both thresholds must be smaller than the Transmit Buffer size of a particular channel.

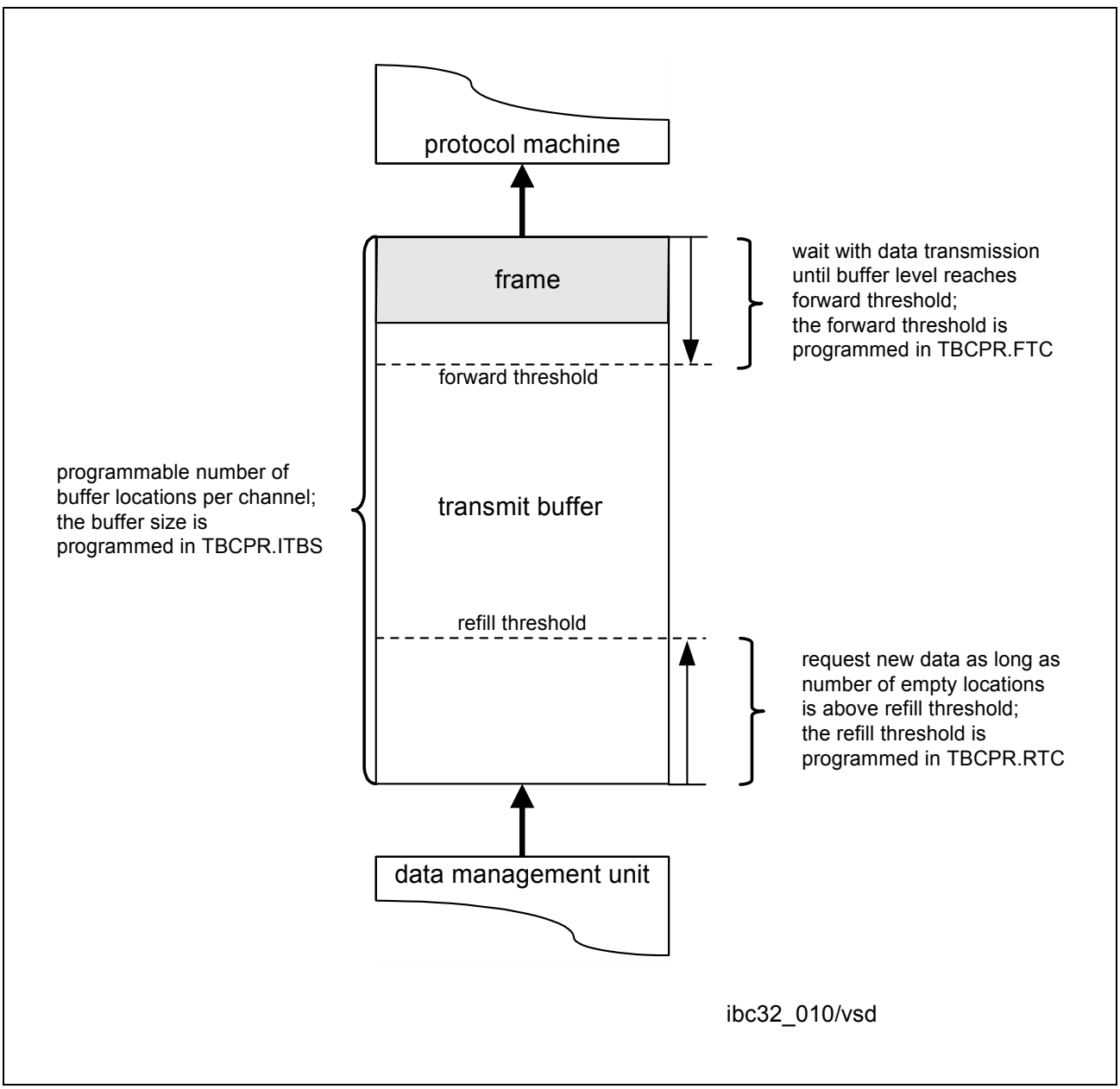

protocol machine
frame
wait with data transmission until buffer level reaches forward threshold; the forward threshold is programmed in TBCPR.FTC
forward threshold
programmable number of buffer locations per channel; the buffer size is programmed in TBCPR.ITBS
transmit buffer
refill threshold
request new data as long as number of empty locations is above refill threshold; the refill threshold is programmed in TBCPR.RTC
data management unit
ibc32_010/vsd

Figure 31-6 Transmit Buffer Thresholds

The threshold values have the following effect:

- Data stored in the internal Transmit Buffer will be transferred to the protocol machine only when the transmit forward threshold is reached or if a complete frame is stored inside the Transmit Buffer. This mechanism avoids data underrun conditions.
- As long as the amount of data stored in the Transmit Buffer is below the transmit refill threshold, the DMU will keep filling the buffer by initiating FPI burst transfers.

Note: Because there is a delay between the time the Transmit Buffer requests data from the DMU and the time the DMU serves the request, the actual number of empty locations may be higher than the transmit refill threshold. To determine the maximum FPI burst length, an additional parameter is available which limits these requests up to a maximum of 8 WORDs (FPI).

31.1.5 MAC Controller

The MAC Controller implements the IEEE 802.3 Specification and operates at either 100 Mbit/s or 10 Mbit/s. In Half Duplex Mode, the controller implements the IEEE 802.3 Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol. In Full Duplex Mode, the controller implements the IEEE 802.3 MAC Control Layer and the Pause Operation for flow control. The controller also supports flow control in Half Duplex Mode, and includes programmable support for additional MAC Control functions.

As part of the evolving support for the Full Duplex Mode of operation, the IEEE 802.3 standards group has approved a new standard for flow control. This standard establishes an optional “MAC Control” sublayer and defines the Pause Operation that is supported by this MAC Control sublayer. The Pause Operation provides an industry standard method of supporting flow control in Full Duplex Ethernet networks.

31.1.5.1 100/10-Mbit/s Ethernet MAC Layer Overview

The 100/10-Mbit/s Ethernet MAC Controller provides the Media Independent Interface (MII). The MII conforms to the International Standard ISO/IEC 8802-3 for a media independent layer which will separate Physical Layer issues from the MAC Layer. The MAC Layer consists of the receive and the transmit blocks, a flow control block, a Content Addressable Memory (CAM) for storing network addresses, and a number of command, status, and error counter registers. The MII supplies the transmit and receive clocks with rates of 25 MHz for 100-Mbit/s operation or 2.5 MHz for 10-Mbit/s operation.

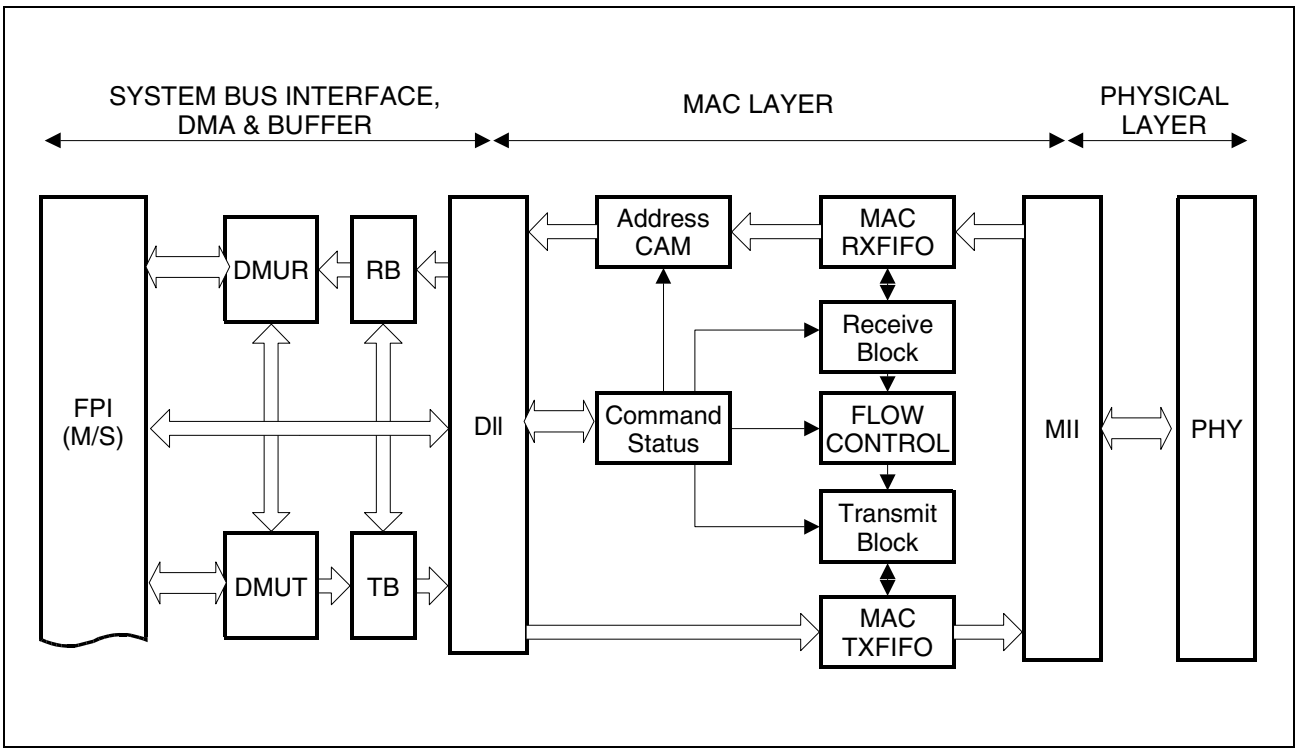


Figure 31-7 Flow Control Ethernet System Overview

31.1.5.2 Functional Blocks Overview

Figure 31-8 shows the major functional blocks of the Flow Control 100/10-Mbit/s Ethernet Stand-alone MAC.

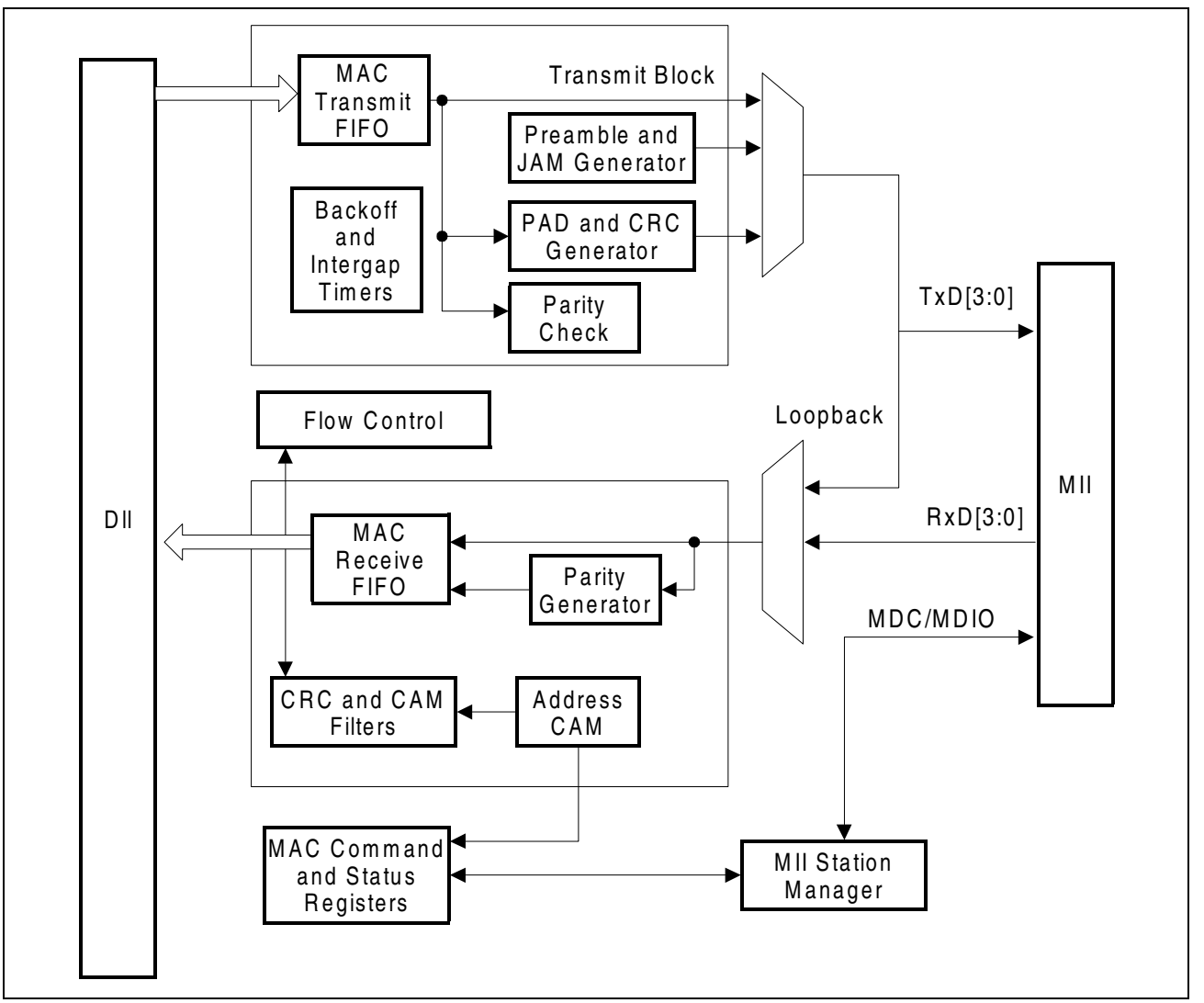


Figure 31-8 MAC Functional Blocks

The Media Independent Interface (MII) is the interface between the Physical Layer and the transmit and receive blocks. The DMA Independent Interface (DII) builds the interface to the on-chip buffers for receive (RB) and transmit direction (TB).

The transmit block moves the outgoing data from the Transmit Buffer to the MII. The transmit block includes circuits for generating the CRC, checking parity, and generating preamble or jam. The transmit block also has timers for backoff after collision and for the interpacket gap after transmission.

The receive block accepts incoming data from the MII and stores it in the receive FIFO. The receive block has logic for computing and checking the CRC value, generating parity for data from the MII, and checking minimum and maximum packet lengths. The receive

block also has a Content Addressable Memory (CAM) block which provides for address lookup and acceptance or rejection of packets based on their destination address.

The flow control block recognizes MAC Control packets and supports the Pause Operation for Full Duplex links. The flow control block also supports generation of Pause packets, and provides timers and counters for pause control.

The command and status registers control programmable options, including the enabling or disabling of signals that notify the system when conditions occur. The status registers hold information for error handling software. An error counter accumulates statistical information for network management software.

The loopback circuit provides the ability to perform MAC Layer testing in isolation from the MII and the Physical Layer.

31.1.5.3 Media Independent Interface (MII)

Both transmit and receive blocks operate using the MII, which was developed by the IEEE 802.3 Task Force on 100 Mbit/s Ethernet. This interface has the following characteristics:

- Media independence
- Multi-vendor point of interoperability
- Supports connection of MAC Layer and Physical (PHY) Layer devices
- Capable of supporting both 100 Mbit/s and 10 Mbit/s data rates
- Data and delimiter are synchronous to clock references
- Provides independent 4-bit wide transmit and receive data paths
- Supports connection of PHY Layer and Station Management (STA) devices
- Provides a simple Management Interface
- Capable of driving a limited length of shielded cable

The Physical Layer entity or PHY performs all of the decoding on the incoming and outgoing data. The manner of decoding (Manchester for 10BASE-T, 4B/5B for 100BASE-X, or 8B/6T for 100BASE-T4) does not affect the MII. The MII provides raw data it receives, starting with the preamble and ending with the CRC. The MII expects raw data for transmission, also starting with the preamble and ending with the CRC. The MAC Layer also generates jam data and transmits it to the PHY.

31.1.5.4 The Transmit Block

The transmit block is responsible for transmitting data. It complies with the IEEE 802.3 Standard for Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol. The transmit block consists of the following sections:

- Transmit FIFO and controllers
- Preamble and jam generators
- Pad generator
- Parallel CRC generator
- Threshold logic and counters
- Back off and retransmit timers
- Transmit state machine
- Flow Control Send Pause

Figure 31-9 shows the functional blocks of the transmit block in more detail.

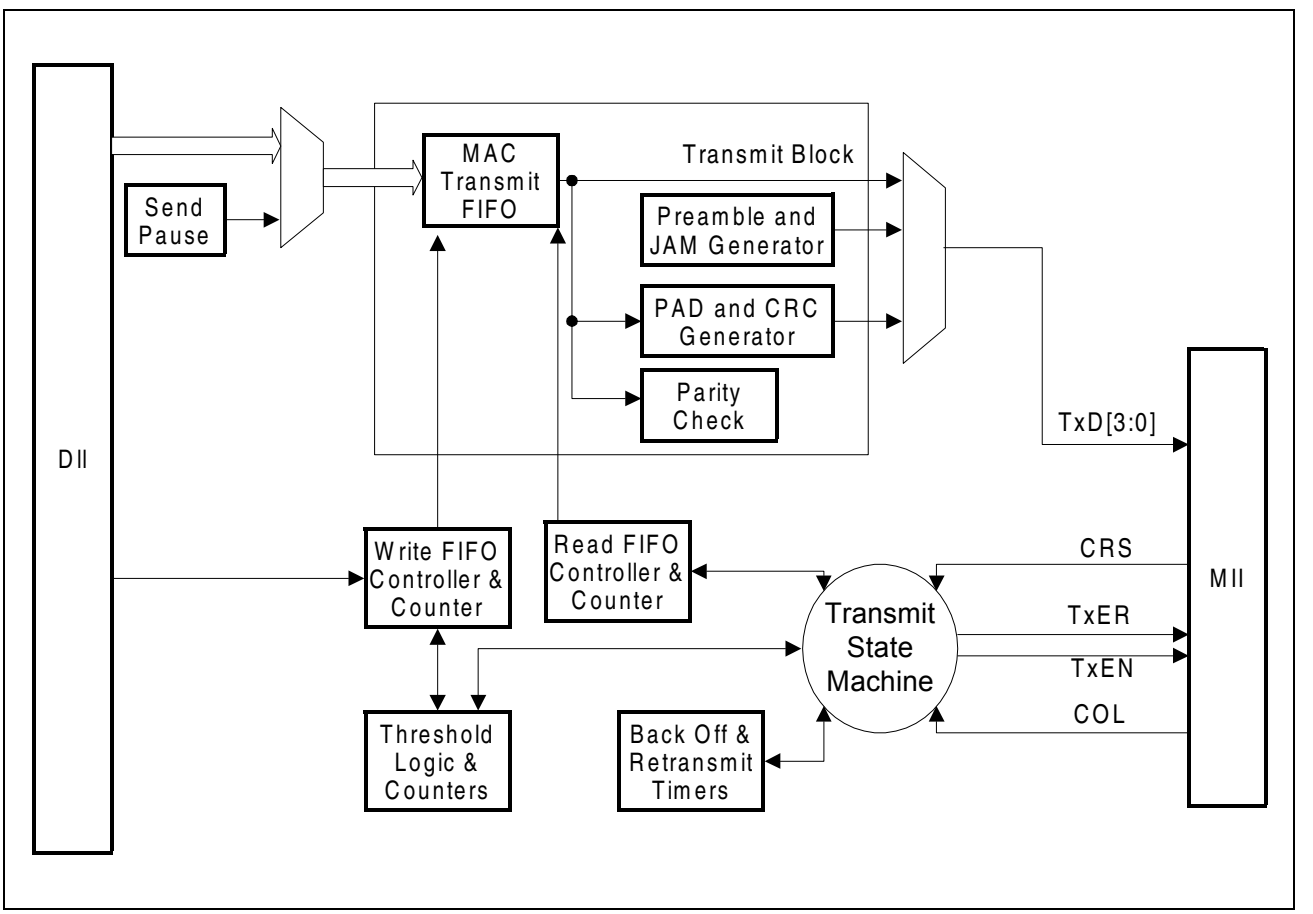


Figure 31-9 Stand-alone MAC Transmit Block

Transmit FIFO and Read/Write Controllers

The transmit FIFO is 80 bytes deep, with an extra bit associated with each data byte for parity. This 80-by-9 size allows the first bytes of a packet to be stored and retransmitted,

without further system involvement, in case of a collision. The additional 16 bytes provide for system latency and avoid FIFO underrun until it is assured that no collision has occurred.

After the system interface sets the transmit enable bit in the control register, the transmit DII state machine requests data from the DII. The FIFO controller stores data in the transmit FIFO until the threshold for transmit data is satisfied. The FIFO controller passes a handshaking signal to the transmit state machine, indicating that sufficient data is in the FIFO to start the transmit operation. The FIFO controller passes on to the DII requests for more data when the FIFO is not full. The transmit state machine will continue transmitting data until it detects the end-of-frame signal, which signals the last byte. It then appends the calculated CRC to the end of the data, unless the CRC truncate bit in the transmit control register is set. Then, the packet transmit bit in the status register is set which can cause an interrupt if enabled.

The FIFO counters in this block (WRITE counter) and the transmit FIFO counter of the transmit state machine (READ counter) coordinate their functions based on each other's count, but they are on different clock domains.

The FIFO controller stores parity with the data in the FIFO. It checks the parity and potentially halts transmission after reading the data out of the FIFO and sending it for the CRC calculation. If a parity error occurs, the FIFO controller sets an error status bit, which can cause an interrupt if enabled.

Preamble and Jam Generator

As soon as the transmit enable bit in the control register is set and there are eight bytes of data in the FIFO, the transmit state machine starts transmission by asserting the TxEN signal and transmitting the preamble and the start frame delimiter (SFD). In case there is a collision, it transmits 32 bits of all 1s after the preamble as a jam pattern.

PAD Generator

If a short data packet is transmitted, the MAC will normally generate pad bytes to extend the packet to a minimum of 64 bytes. The pad bytes have bits of all 0s. There is a control bit to suppress the generation of pad bytes.

Parallel CRC Generator

The CRC generation of the outgoing data starts from the Destination Address and continues through the data field. CRC generation can be suppressed, by setting a bit in the transmit control register. This is primarily useful in testing, e.g. to force the transmission of a bad CRC and test error detection in the receiver. It can also be useful in certain bridge or switch applications, where end-to-end CRC checking is desired.

Threshold Logic and Counters

The transmit state machine uses a counter and logic to control the threshold for beginning a transmission. The MAC will wait until eight bytes have been placed in the Transmit FIFO before attempting to initiate transmission. This provides the DMA engine some latency without causing underflow during transmission.

Back Off and Retransmit Timers

When a collision is detected on the network, the transmitter block stops the transmission and starts a jamming pattern to ensure that all the nodes detect the collision. After this, the transmitter waits for a minimum of 96 bit times and then retransmits the data. After 16 attempts, the transmit state machine sets an error bit and generates an interrupt, if enabled, to signify the failure to transmit a packet due to excessive collisions. It flushes the FIFO, and the MAC is ready for the next packet.

Transmit Data Parity Checker

Data in the FIFO is even-parity protected. When data is read for transmission, the transmit state machine checks the parity and in the case of an error:

- It stops transmission.
- It sets the parity error bit in the transmit status register.
- It generates an interrupt if enabled.

Transmit State Machine

This is the central control logic for the transmit block. It controls the passing of signals, the timers, and the posting of errors in the status registers.

31.1.5.5 Receive Block

The receive block is responsible for receiving data. It complies with the IEEE 802.3 Standard for Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol.

After receiving a packet, the receive block checks for a number of error conditions: CRC error, alignment error, and length errors. Several of these can be disabled by setting bits in the control registers. Depending upon the CAM status, the destination address, and signals from an external CAM, the receive block may reject an otherwise acceptable packet. The receive block consists of the following sections:

- Receive FIFO, FIFO controller, and counters
- Receive DII state machine
- Threshold logic and counters
- CAM block for address recognition
- Parallel CRC checker
- Receive state machine
- MAC Control and Pause

The functional blocks of the receive block are shown in more detail in **Figure 31-10**.

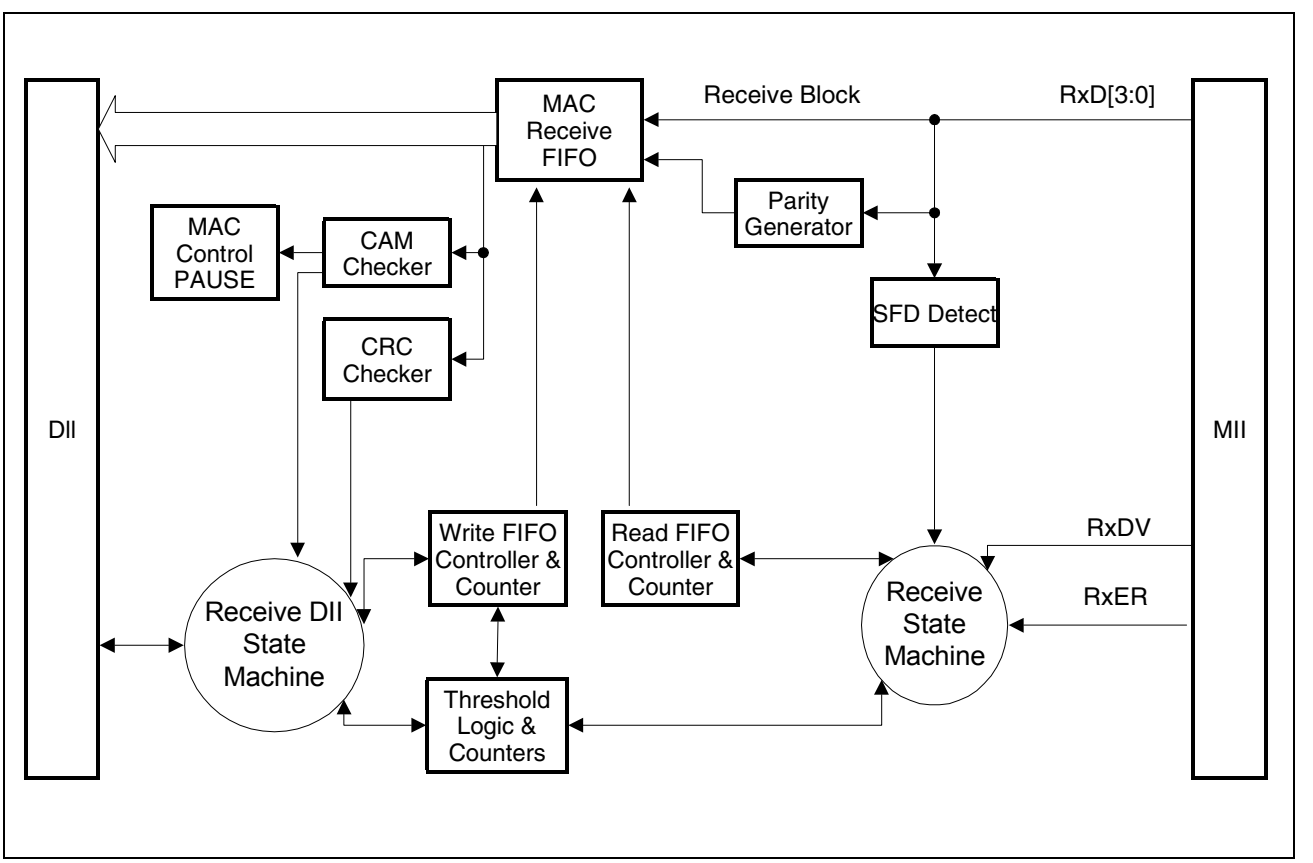


Figure 31-10 Stand-alone MAC Receive Block

Receive FIFO Controller

The receive FIFO accepts data one byte at a time with parity, starting with the destination address. It updates the counter with the number of bytes received. As the FIFO stores the data, the CAM block checks the destination address against its stored addresses, and if it recognizes it, the FIFO will continue receiving the packet. The received data bytes are forwarded to an interface FIFO towards the receive buffer of 64 bytes. If the CAM block rejects the packet, the receive block discards it and flushes the FIFO.

Address CAM and Address Recognition

The CAM block provides direct-compare address recognition. The CAM compares the destination address of the received packet to pre-stored addresses. When it finds a match, the receive state machine continues to receive the packet.

The CAM is organized to hold 6-byte address entries. The standard CAM size for the Flow Control MAC is 22 entries.

Parallel CRC Checker

The receive block computes a CRC across the data and the transmitted CRC and checks that the resulting syndrome is valid. A parallel CRC checking scheme is used to handle data arriving at 100 Mbit/s. The receive and transmit blocks have independent CRC circuits to support Full Duplex operation.

Receive State Machine

In MII mode, the receive block receives data on the RxD[3:0] lines from the MII, synchronous to Rx_clk at 25 MHz or 2.5 MHz. After detecting the preamble and SFD, it arranges data in byte configurations, generates parity, and stores the result in the receive FIFO one byte at a time. At the end of the reception, the receive block marks the packet received by setting the appropriate bits in the receive status register. Any error in reception will reset the FIFO and the state machine will wait for the end of the current packet, and then idle looking for the next preamble and SFD.

Receive DII State Machine

The receive DII state machine signals to request emptying the receive FIFO whenever there is data in the receive FIFO.

31.1.5.6 Flow Control Block

The flow control block provides the following functions:

- Recognition of MAC Control frames received by the receive block
- Transmission of MAC Control frames, even if transmitter is paused
- Timers and counters for Pause operation
- Command and Status Register interface
- Options for passing MAC Control frames through to software drivers

The receive logic in the flow control block recognizes a MAC Control frame as follows: First, the length/type field must have the special value specified for MAC Control frames. Second, the destination address must be recognized by the CAM. Third, the frame length must be 64 bytes, including CRC. Fourth, the CRC must be good. And fifth, the frame must contain a valid Pause opcode and operation.

If the length/type field does not have the special value specified for MAC Control frames, then the MAC takes no action, and the packet is treated as a normal packet. If the CAM does not recognize the destination address, the MAC rejects the packet. If the packet length is not 64 bytes, including CRC, the MAC will not perform the operation. The packet will be marked as a MAC Control packet, and passed forward to the software drivers, if pass through is enabled.

A control bit in the Transmit Status register can be set to generate a Full Duplex Pause Operation or other MAC Control function, even if the transmitter itself is paused.

There are two timers and corresponding CSR registers that are used during Pause operation. One timer and register are used when a received packet causes the transmitter to be paused. The other pair is used to approximate the pause status of the other end of the link, after the transmitter sends a Pause command.

The Command and Status Register (CSR) interface provides control and status bits within the transmit and receive control registers and status registers. These allow the initiation of sending a MAC Control frame, enabling and disabling MAC Control functions, and reading of the Flow Control counters.

Control bits are provided for either processing MAC Control frames entirely within the controller, or for passing MAC Control frames on to the software drivers. This allows Flow Control to be enabled by default even on software drivers which are not otherwise “Flow Control aware”.

31.1.5.7 Detailed Operation

This section gives detailed information about the following aspects of MAC operation:

- MAC frame and packet formats
- Initialization
- Register access
- Transmitting a frame
- Receiving a frame
- CAM Operation
- Full Duplex Pause Operation
- Error signaling and network management
- Accessing Station Management

MAC Frame and Packet Formats

Table 31-6 shows the format of an IEEE 802.3/Ethernet frame. The standard packet has the following fields:

- Preamble – seven identical bytes. The bits in each byte are 10101010, transmitted from left to right.
- Start Frame Delimiter (SFD) – one byte. The bits are 10101011, transmitted from left to right.
- Destination Address – six bytes. May be an individual or a multicast address. The CAM provides address filtering using the Destination Address.
- Source Address – six bytes. MAC does not interpret the source address.
- Length or Type Field – two bytes. The MAC only recognizes the special value of 8808h as the MAC Control Frame Type. Other values are not processed. The current IEEE 802.3 Standard specifies that values less than 1500 are lengths and values greater than 1535 are types. Values less than or equal to 1500 indicate the number of Logical Link Control (LLC) data bytes in the data field. The MAC transmits the high-order byte of the Length or Type Field first.
- Logical Link Control (LLC) Data – 46 to 1500 bytes.
- Pad – 0 to 46 bytes. If the user-provided LLC Data is less than 46 bytes long, the MAC will transmit pad bytes of 0s.
- Cyclic Redundancy Check (CRC) – four bytes. A value computed as a function of all fields except the preamble, the SFD, and the CRC itself. The IEEE 802.3 standard also refers to the CRC as the Frame Check Sequence (FCS).

Table 31-6 Fields of an IEEE 802.3/Ethernet Packet (Frame)

Packet (encoded on the medium)											
Added by transmitter,		**Data Frame (sent by user)**					**Added by Transmitter**				
Stripped by receiver		**Data Frame (delivered to user)**									
Preamble	SFD	Destination address	Source address	Length or Type			**Optionally stripped by Receiver**				
						LLC Data		**CRC (FCS)**			
				high	low	User Data	Pad Data	MSB			LSB
7 bytes	1	6 bytes	6 bytes	2		0-1500	0-46	4 bytes			

The Preamble, SFD, Pad Data, and CRC are added by the transmitter. Padding can also be done in software, and there is a Transmit Control bit to suppress CRC addition. The Receive Control register has a bit to control stripping the CRC. Stripping of Pad Data is the responsibility of the software drivers.

The MAC transmits the Least Significant Bit of each byte first for all fields except the CRC. Throughout this document, we attempt to use “packet” to denote all of the bytes transmitted and received, while “frame” refers to the bytes delivered by the user for transmission, and to the user who is receiving.

There are a number of factors and options which can affect this “standard MAC frame”:

- Some PHYs may deliver a longer or shorter preamble.
- Short Packet Mode allows LLC data fields with less than 46 bytes. There are options to suppress padding and allow reception of short packets. But in the TC1130, a frame less than 64 bytes is not transmitted.
- Long Packet Mode allows LLC data fields with more than 1500 bytes. There is an option to allow reception of long packets.
- No CRC Mode suppresses the appending of an CRC field.
- Ignore CRC Mode allows the reception of packets without valid CRC fields.

Destination Address Format

Bit 0 of the destination address is an address type designation bit. It identifies the address as either an individual or a group address. Group addresses are also called multicast addresses. Individual addresses are also called unicast addresses. The broadcast address is a special group address, namely FF-FF-FF-FF-FF-FF in hexadecimal format.

Bit 1 distinguishes between locally or globally administered addresses. For globally administered (or U, universal) addresses, the bit is set to 0. If an address is to be assigned locally, this bit is set to 1. For the broadcast address, this bit is also a 1.

Destination Address, First Byte

Field	Bits	Type	Description
I/G	0	–	**Individual or Group Flag** 0 Individual address 1 Group address
U/L	1	–	**Universal or Local Flag** 0 Universal address 1 Local address
Rest	[7:2]	–	**Rest of Byte** Rest of first byte of destination address

Special Flow Control Destination Address

The current specification for Full Duplex Flow Control specifies a special destination address for the Pause Operation packet. In order for the MAC to receive packets which contain this special destination address, the address must be programmed into one of the CAM entries, that CAM entry must be enabled, and the CAM must be activated.

Some CAM entries are also used when generating a Flow Control packet via the SdPause bit in the Transmit Control register, as explained later.

Initialization

On power up and reset, the Flow Control and MAC control and status registers are set. Transmit Collision Count, CAM Data registers are not set on power up or reset. The Transmit Collision Count register is reset at the beginning of transmitting a new packet. The CAM memory should be initialized before enabling the use of the CAM circuit.

Transmitting a Frame

To transmit a frame, the transmit enable bit in the transmit control register must be set and the transmit halt request bit must be zero. In addition, the halt immediate and halt request bits in the MAC Control register must be zero. These conditions are normally set after any DMA controller initialization has occurred.

The transmit state machine will start transmitting the data in the FIFO, and will retain the first 64 bytes until after this station has acquired the net. At that time, the transmit block will request more data and transmit it until signaling the end of data to be transmitted. The transmit block appends the calculated CRC to the end of the packet, and transmission ends. It sets transmit status register Bit 0, signaling a successful transmission, which in turn may cause an interrupt.

The transmit operation can be divided into two operations, the MII interface and the DII interface.

- The DII transmit operation

The DII transmit operation is a simple FIFO mechanism. The DMA engine stores data to be transmitted, and the transmit state machine empties it when the MAC successfully acquires the net. Note that the two time domains intersect at the FIFO controller. The writing and reading of data is asynchronous and on different clocks. Reading is driven by either a 25 MHz or 2.5 MHz transmit clock. Writing is driven by the synchronous Sys_clk, which is asynchronous to Tx_clk.

After reset, the transmit FIFO is empty, the transmit block asserts the Tx_rdy signal, and transmission is disabled. To enable transmission, the system must set the transmit enable bit in the Transmit Control register. Also, there must be eight bytes of data in the transmit FIFO. The DMA engine can start stuffing data into the FIFO, and then enable the transmit bit, or vise versa. The transmit operation only starts if both of these conditions are met.

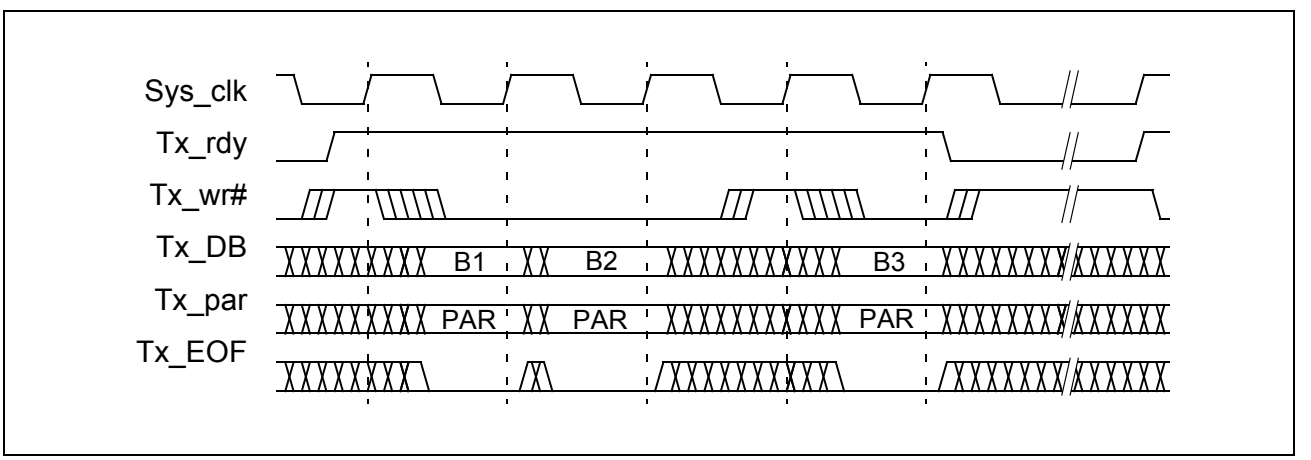


Figure 31-11 DII Transmit Data Timing

When the transmit block asserts the Tx_rdy signal, the DMA engine can write data into the transmit FIFO by asserting the Tx_wr# signal. **Figure 31-11** shows timing sequences for back-to-back transfers or transfers with wait states. This is a synchronous interface, and data is latched in at the rising edge of the Sys_clk when Tx_wr# is asserted. For slower interfaces the rising edge of Tx_wr# can be delayed. This is the equivalent of asserting wait states in a synchronous operation. The transmit FIFO machine will check the Tx_par and the Tx_EOF bits. If there is a parity error, the transmit block aborts the

transmission, resets the FIFO, and generates an interrupt by setting the TxPar bit in the transmit status register. The Tx_EOF bit signals the end of one frame to be transmitted.

- The MII transmit operation

The transmit block consists of three state machines. The gap_ok state machine tracks and counts the intergap timing between the frames. In non-full duplex mode, it counts 96 bit times from the deassertion of the ECRS signal. If there is any traffic within the first 64 bit times the gap_ok state machine resets and starts counting from zero. If there is any traffic in the last 1/3 of the interframe gap, then it continues counting and sends a gap_ok signal to the main transmit state machine. Once a packet has been transmitted successfully, gap_ok will be sent out at the end of the next 96 bit times regardless of the network traffic. In full duplex mode, the state machine starts counting at the end of transmission and gap_ok will be sent at the end of the 96 bit times regardless of the network traffic.

The back_off state machine implements the back off and retry algorithm of the 802.3 CSMA/CD. Upon the detection of a collision, the main state machine starts the back_off state machine counters and waits for the end of the back off time (including zero), which is a multiple of 512 bit times, before retransmitting the collision-causing packet again. Each time there is a collision (for one single packet), back_off state machine increments an internal attempt counter. A 11-bit pseudo random number generator outputs a random number by selecting a subset of the value of the generator at any time. The subset grows by one bit for each subsequent attempt. This implements the equation:

$$0 \leq r < 2^k, \text{ with } k = \min(n, 10) \qquad (31.1)$$

where r is the number of slot times that the MAC has to wait in case of a collision, and n is the number of attempts. For example, after the first collision, n is 1 and r is a random number between 0 and 1. The pseudo random generator in this case is one-bit wide and gives a random number of either 0 or 1. After the second attempt, r is a random number between 0 and 3; so the state machine looks at the two least significant bits of the random generator (n = 2) which gives a value between 0 and 3.

In order to improve the statistical independence between two MACs using the same pseudo-random number generator, the MAC uses values from the CRC of previous successfully transmitted packets to modify the basic random number sequence.

The main transmit state machine implements the rest of the MAC layer protocols. If there is data to be transferred, and the interframe gap is OK, and the MII is ready (there are no collisions and if not in duplex mode there is no ECRS), then the transmit block transmits the preamble followed by the SFD. After the transmission of the SFD and the preamble, it transmits 64 bytes of the data regardless of the packet length, unless short transmission is enabled. This means that if the packet is less than 64 bytes, it will pad the LLC data field with zeros, and then appends the CRC to the end of the packet, if CRC generation is enabled. If there is any collision during this first 64 bytes (8 bytes of preamble and SFD and 56 bytes of the frame), it stops the transmission and transmits a

jam pattern (32 bits of all ones). It increments the collision attempt count, returns control to the back_off state machine, and retransmits the packet when the back off time has elapsed and the gap time is OK.

If there are no collisions, the transmit block transmits the rest of the packet, and at this time (after the first 60 bytes have been transmitted without collisions), it allows the DMA engine to overwrite this packet. After transmitting the first 64 bytes, the transmit block transmits the rest of the packet and appends the CRC to the end. Parity errors, FIFO errors, or more than 16 collisions will cause the state machine to abort the packet (no retry) and queue up the next packet.

In case of any transmission errors, the transmit block sets the appropriate error bit in the transmit status register. It may also generate an interrupt, depending on the enable bits in the interrupt mask register.

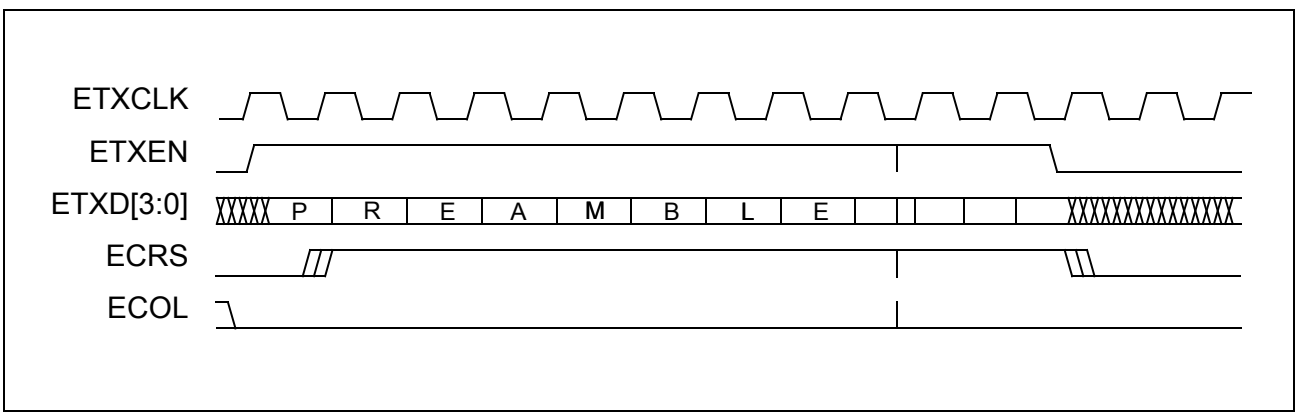


Figure 31-12 Transmission Without Collision

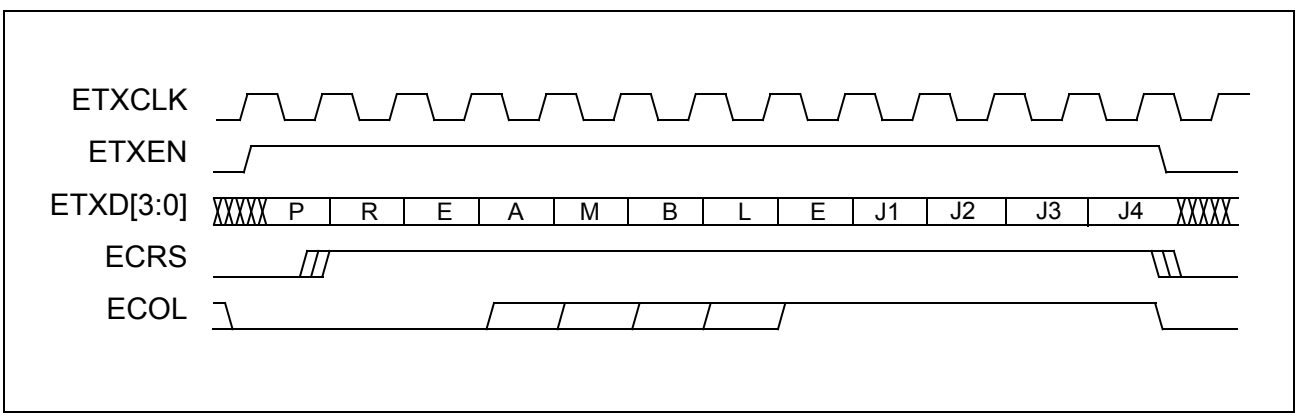


Figure 31-13 Transmission With Collision in Preamble

Receiving a Frame

The receive block, when enabled, constantly monitors a data stream coming either from the MII or, if in loop back mode, from the transmit block. The MII supplies from 0 to 7 bytes of preamble, followed by the Start Frame Delimiter (SFD). The receive block checks that the first nibbles received are preamble, and looks for SFD (10101011) in the

first 8 bytes. If the SFD does not appear by then, it treats the packet as a fragment and discards it.

The first nibble of destination address follows the SFD, Least Significant Bits first. When it has received a byte, the receive block forwards it to the on-chip data buffer. When receiving data, CRC is calculated in parallel and is checked with the CRC at the end of the frame.

It is the responsibility of the DMUR to empty the receive buffer.

CAM Operation

The CPU must configure the CAM. Therefore, it must write the 32-bit data into the register MACCAMDAT and the corresponding CAM location (WORD address) into the register MACCAMADDR. Afterwards, the MAC sets the bit MACCAMADDR.CAW and stores the data in the internal CAM. As soon as the bit MACCAMADDR.CAW has been reset again by the MAC, the CPU is allowed to enter additional data.

Table 31-7 shows how the MAC reads CAM entries from the CAM memory. Entries are assumed to be in Big Endian order: #0-0 is the first byte of the first entry, #0-5 is the sixth and last byte of the first entry, and so on. There are two bytes after CAM entry #20 (Rsv-2 and Rsv-3), and three double-words (MC#1, MC#2, and MC#3), which are not used in CAM operation, but are used in generating MAC Control Frames, as explained in following section.

Table 31-7 CAM Memory Map

Byte 3	Byte2	Byte 1	Byte 0	8-bit Address	32-bit Address
#0-0	#0-1	#0-2	#0-3	00	00
#0-4	#0-5	#1-0	#1-1	04	01
#1-2	#1-3	#1-4	#1-5	08	02
#2-0	#2-1	#2-2	#2-3	0C	03
#2-4	#2-5	#3-0	#3-1	10	04
#3-2	#3-3	#3-4	#3-5	14	05
#4-0	#4-1	#4-2	#4-3	18	06
#4-4	#4-5	#5-0	#5-1	1C	07
#5-2	#5-3	#5-4	#5-5	20	08
				…	…
#18-0	#18-1	#18-2	#18-3	6C	1B
#18-4	#18-5	#19-0	#19-1	70	1C
#19-2	#19-3	#19-4	#19-5	74	1D

Table 31-7 CAM Memory Map (cont'd)

Byte 3	Byte2	Byte 1	Byte 0	8-bit Address	32-bit Address
#20-0	#20-1	#20-2	#20-3	78	1E
#20-4	#20-5	Rsv-2	Rsv-3	7C	1F
MC#1-0	MC#1-1	MC#1-2	MC#1-3	80	20
MC#2-0	MC#2-1	MC#2-2	MC#2-3	84	21
MC#3-0	MC#3-1	MC#3-2	MC#3-3	88	22

The CAM memory contains thirty-five 32-bit words providing space for

- Up to twenty-one 6-byte addresses
- Two reserved bytes
- Three double-words (each of 4 bytes) for Pause operation

Full Duplex Pause Operation

Transmit Pause Operation

To enable Full Duplex Pause Operation, the Special Broadcast address for MAC Control Packets must be programmed into the CAM, and the corresponding CAM Enable bit set. While this can be any CAM location, the next section will specify how some CAM locations may be preferred, to optimize CAM entry utilization.

The MAC Receive circuit recognizes the Full Duplex Pause Operation when the following conditions are met:

- The Type/Length field has the Special value for MAC Control packets, 0x8808
- The packet is recognized by the CAM
- The length of the packet is 64 bytes
- The operation field specifies Pause operation

When a Full Duplex Pause Operation is recognized, the MAC Receive circuit loads the operand value into the Pause Count Register, and signals both the MAC and the external DMA engine that pause should begin at the end of the current packet, if any.

The Pause circuit maintains the Pause Counter, and decrements it to zero, before signaling the end of the pause operation, and allowing the Transmit circuit to resume.

If a second Full Duplex Pause Operation is recognized while the first operation is in effect, the Pause Counter is reset with the current operand value. Note that a value of 0 may cause pre-mature termination of a pause operation in progress.

Remote Pause Operation

To send a remote Pause Operation the following steps need to be taken:

1. Program CAM location #0 with the Destination Address
2. Program CAM location #1 with the Source Address
3. Program CAM location #20 length/type Field, opcode, and operand
4. Program the 2 bytes after operand with 0000_H
5. Program the three double-word after CAM location #20 with zeros
6. Write the Transmit Control Register, setting the SdPause bit

The Destination Address and Source Address are normally the Special Broadcast Address for MAC Control Frames and the local Station Address, respectively. The Flow Control 100/10-Mbit/s Ethernet Stand-alone MAC supports full programmability of these values to support future uses of MAC Control frames.

Upon completion, the transmit status is available through the MACTXSTAT register. An interrupt is generated, if enabled by the MACxIMR register.

Error Signaling

The error/abnormal operation flags asserted by the MAC are arranged into transmit and receive groups and can be found in either Transmit Status Register (MACTXSTAT) or Receive Status Register (MACRXSTAT). Additionally, there is a missed packet error counter for system network management purposes.

Any of these error status information can trigger an interrupt. Two interrupt status registers are provided: MACISR0 and MACISR1. Interrupts can be enabled/disabled in the corresponding interrupt mask registers MACIMR0 and MACIMR1.

Reporting of Errors in Transmit

Transmit operation terminates when the entire packet (preamble, SFD, data, and CRC) has been successfully transmitted through MII without encountering a collision. Additionally, the transmit block detects and reports both internal and network errors.

Under the following conditions, the operation will be aborted (in most cases).

Table 31-8 Transmit Errors

Parity Error	The 8 bits of data coming in through the DII has an optional parity bit. Also a parity bit protects each byte in the MAC transmit FIFO. If a parity error occurs, it will be reported to the transmit state machine, and the transmission will be aborted. A detected parity error sets the TxPar bit in the Transmit Status Register.
Transmit FIFO Underrun	The 80-byte transmit FIFO provides 16 bytes or 128 bit time for system latency (1.28 µs for 100 Mbit/s, 12.8 µs for 10 Mbit/s). An underrun transmit FIFO during transmission indicates a system problem (the system cannot keep up with the demand of the MAC), and the transmission will be aborted. Additionally, up to 1024 WORDs can be stored in the Transmit Buffer (TB) to allow a much higher latency time.
Lost CRS	The Carrier Sense (ECRS) is monitored from the beginning of the Start of Frame Delimiter (SFD) to the last byte transmitted. A Lost ECRS indicates that ECRS was never present or was dropped during transmission (a possible network problem), but the transmission will NOT be aborted. During loopback mode, the MAC is disconnected from the network, and No ECRS will not be detected.
Excessive Collision Error	Whenever the MAC encounters a collision during transmit, it will backoff, update the “attempt counter”, and retry again in the later time. When the counter reaches the value of 16 (16 attempts all result in a collision), the transmission will be aborted. It indicates a network problem.
Late Collision Error (Transmit Out Of Window Collision)	Normally, the MAC would see a collision (if there is one) within the first 64 bytes of data being transmitted which includes the preamble and SFD. If in any case a collision comes in beyond this time frame a possible network problem is detected. The error will be reported to the transmit state machine, and the transmission will be aborted.
Excessive Deferral Error	During the first attempt of sending a packet, the MAC may have to defer the transmission because of pre-occupied network. If the deferral time is longer than a certain amount (24 kbit time) the transmission will be aborted. The excessive deferral indicates a possible network problem.

Reporting of Errors in Receive

The receive state machine starts putting received data from MII into the receive FIFO after detecting the Start of Frame Delimiter (SFD), and it also checks the internal error (FIFO Overrun) during reception. At the end of reception, the receive state machine looks for external errors (Alignment, Length, CRC, and Frame Too Long).

Table 31-9 Receive Errors

Parity Error	A parity bit protects each byte in the MAC receive FIFO. If a parity error occurs it will be reported to the receive state machine. A detected parity error sets the RxPar bit in the Receive Status Register.
Frame Alignment Error (Dribble)	At the end of reception, the receive block checks that the incoming packet (including CRC) has been correctly framed on an 8-bit boundary. If it is not and the CRC is invalid, data has been disrupted through the network, and the receive block reports a frame alignment error. A CRC error is also reported.
CRC Error	At the end of reception, the receive block checks the CRC for validity, and reports a CRC error if it is invalid. CRC, Frame Alignment, and Length errors are the network errors detected by the receive unit. They might be detected in one of the following combinations: CRC error only Frame Alignment and CRC errors only Length and CRC errors only Frame Alignment, Length, and CRC errors
Frame Too Long	The receive block checks the length of the incoming packet at the end of reception (including CRC, but excluding preamble and SFD). If the length is longer than the maximum frame size of 1518 bytes, the receive block reports receiving a “Long Packet”, unless long frame mode is enabled.
Receive FIFO Overrun	During reception, the incoming data are put into the receive FIFO temporarily before they are transferred to the system memory. If the FIFO is filled up because of excessive system latency or other reasons, the receive block sets the overrun bit in the Receive Status register.
MII Error	The PHY will inform the MAC if it detects a medium error (such as coding violation) by asserting the input pin Rx_er. When the MAC sees ERXER asserted, it sets CRCE bit of the Receive Status register.

31.1.6 Interrupt Handling

The Ethernet Controller supports the generation of nine Interrupts grouped as follows:

- MAC Controller Transmit Interrupt (MACTX0, MACTX1)
  - Excessive Collision, Transmit Deferred, Paused, Underrun, Excessive Deferral, Lost Carrier, Late Collision, Transmit Parity, Completion, Transmit Halted, Signal Quality Error
- MAC Controller Receive Interrupt (MACRX0, MACRX1)
  - Missed Error Counter Roll, Control Frame Received, ALignment Error, CRC Error, Overflow, Long Error, Receive Parity Error, Good, Receive Halted
- DMUR Interrupt (DR)
  - Command Completed, Silent Discard, Invalid Length, CRC Error, Receive Frame Overflow, Maximum Frame Length, Hold Caused Receive Abort, Frame End, Receive Abort, Host Initiated
- DMUT Interrupt (DT)
  - Command Completed, Hold Caused Transmit Abort, Transmit Abort, Host Initiated
- RB Interrupt (RB0, RB1)
  - Data Buffer Free Pool interrupt, Action Queue Free Pool interrupt
- TB Interrupt (TB)
  - Command Failed

The Interrupt structure of the Ethernet Controller is shown in both **Figure 31-14** and **Figure 31-15**. The interrupt request flags, located in the status registers, are set by hardware. The interrupt request flags can be reset via reading these registers. Both **Figure 31-14** and **Figure 31-15** also show the assignment of the interrupt enable flags.

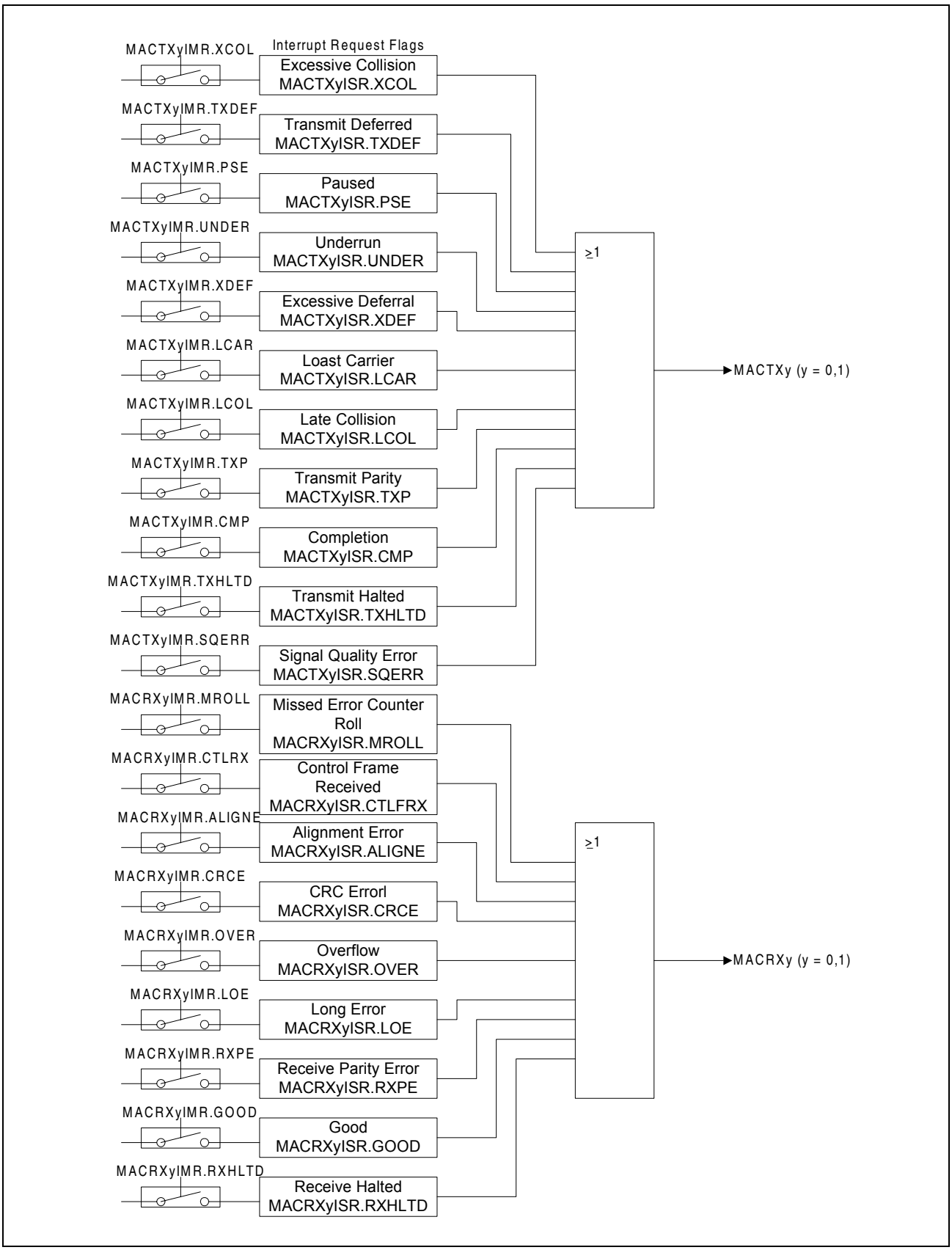


Figure 31-14 Interrupt Structure of Ethernet MAC Controller

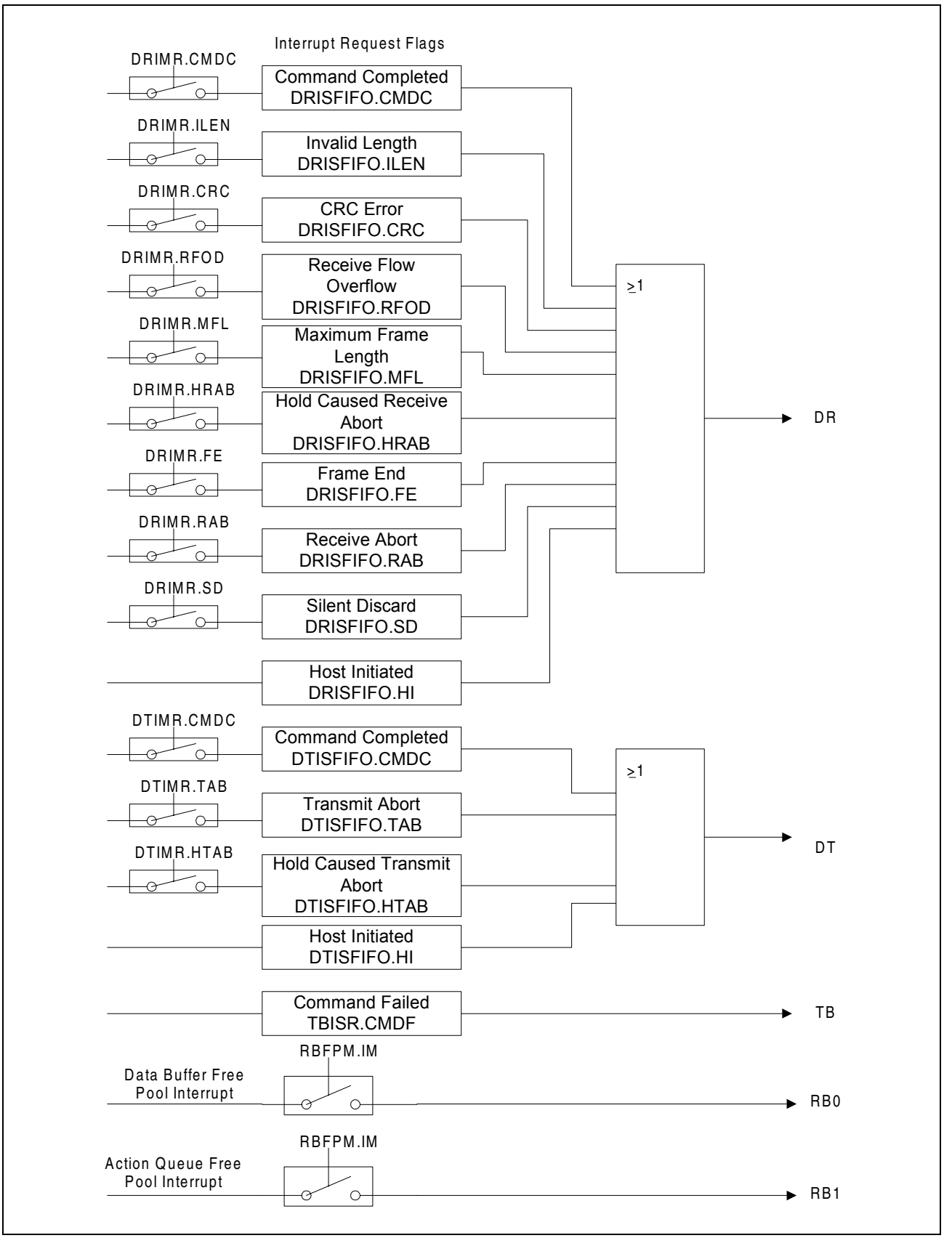


Figure 31-15 Interrupt Structure of Ethernet Controller DMUR, DMUT, TB and RB

31.2 Ethernet Controller Kernel Registers

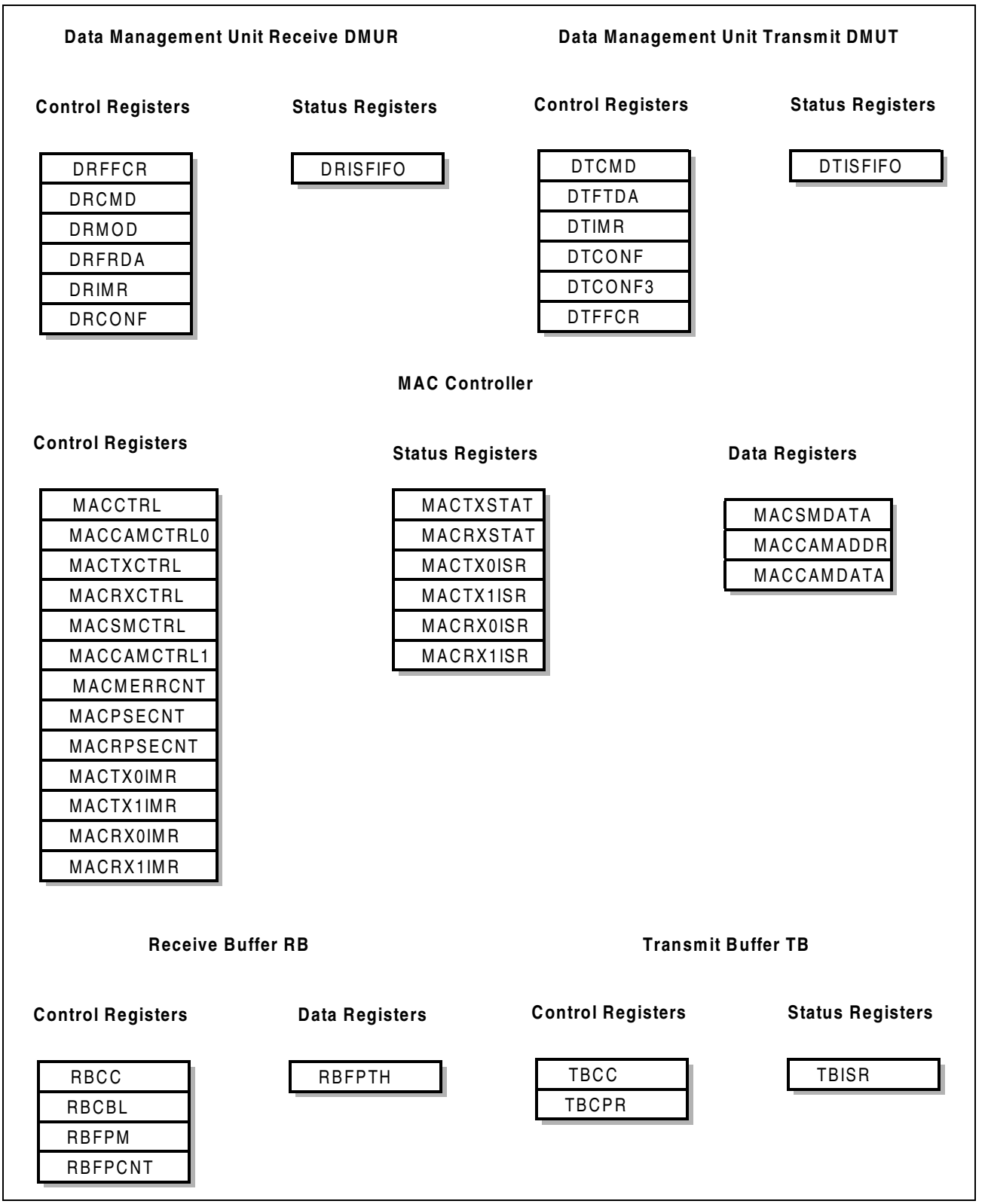


Figure 31-16 Ethernet Controller Kernel Registers

Table 31-10 Ethernet Controller Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
DMUR Unit			
DRISFIFO	DR Interrupt Status FIFO	0018_H	Page 31-82
DRFFCR	DR FIFO Full Counter Register	$001C_H$	Page 31-85
DRCMD	DR Command Register	0020_H	Page 31-86
DRMOD	DR Mode Register	0024_H	Page 31-88
DRFRDA	DR First Receive Descriptor Address Register	$002C_H$	Page 31-88
DRIMR	DR Interrupt Mask Register	0030_H	Page 31-89
DRCONF	DR Configuration Register	0034_H	Page 31-90
DMUT Unit			
DTCMD	DT Command Register	0118_H	Page 31-91
DTFTDA	DT First Transmit Descriptor Address Register	0120_H	Page 31-93
DTIMR	DT Interrupt Mask Register	0124_H	Page 31-94
DTCONF	DT Configuration Register	0128_H	Page 31-95
DTCONF3	DT Configuration 3 Register	$012C_H$	Page 31-95
DTISFIFO	DT Interrupt Status FIFO Register	0138_H	Page 31-96
DTFFCR	DT FIFO Full Counter Register	$013C_H$	Page 31-98
MAC Controller			
MACCTRL	MAC Control Register	0210_H	Page 31-52
MACCAMCTRL0	MAC CAM Control Register 0	0214_H	Page 31-53
MACTXCTRL	MAC Transmit Control Register	0218_H	Page 31-54
MACTXSTAT	MAC Transmit Status Register	$021C_H$	Page 31-55
MACRXCTRL	MAC Receive Control Register	0220_H	Page 31-57
MACRXSTAT	MAC Receive Status Register	0224_H	Page 31-58
MACSMDATA	MAC Station Management Data Register	0228_H	Page 31-59
MACSMCTRL	MAC Station Management Control Register	$022C_H$	Page 31-60
MACCAMADDR	MAC CAM Address Register	0230_H	Page 31-61

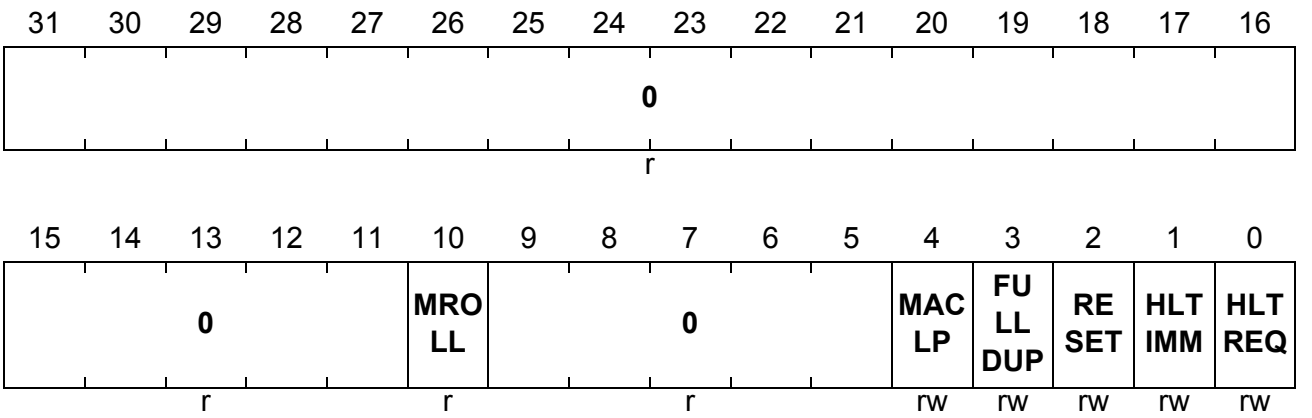
Table 31-10 Ethernet Controller Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
MACCAMDATA	MAC CAM Data Register	0234_H	Page 31-61
MACCAMCTRL1	MAC CAM Control Register 1	0238_H	Page 31-62
MACMERRCNT	MAC Missed Error Count Register	$023C_H$	Page 31-63
MACPSECNT	MAC Pause Count Register	0240_H	Page 31-64
MACRPSECNT	MAC Remote Pause Count Register	0244_H	Page 31-65
MACTX0IMR	MAC Transmit 0 Interrupt Mask Register	0248_H	Page 31-66
MACTX0ISR	MAC Transmit 0 Interrupt Status Register	$024C_H$	Page 31-68
MACTX1IMR	MAC Transmit 1 Interrupt Mask Register	0250_H	Page 31-70
MACTX1ISR	MAC Transmit 1 Interrupt Status Register	0254_H	Page 31-72
MACRX0IMR	MAC Receive 0 Interrupt Mask Register	0268_H	Page 31-76
MACRX0ISR	MAC Receive 0 Interrupt Status Register	$026C_H$	Page 31-78
MACRX1IMR	MAC Receive 1 Interrupt Mask Register	0270_H	Page 31-80
MACRX1ISR	MAC Receive 1 Interrupt Status Register	0274_H	Page 31-78
RB			
RBCC	RB Channel Command Register	0318_H	Page 31-99
RBCBL	RB Channel Burst Length Register	$031C_H$	Page 31-100
RBFPM	RB Free Pool Monitor Register	0320_H	Page 31-101
RBFPTH	RB Free Pool Threshold Register	0324_H	Page 31-102
RBFPCNT	RB Free Pool Count Register	0330_H	Page 31-103
TB			
TBISR	TB Interrupt Status Register	0414_H	Page 31-104
TBCC	TB Channel Command Register	0418_H	Page 31-105
TBCPR	TB Channel Parameter Register	$041C_H$	Page 31-106

31.2.1 Flow Control 100/10 Mbit/s Ethernet MAC Registers

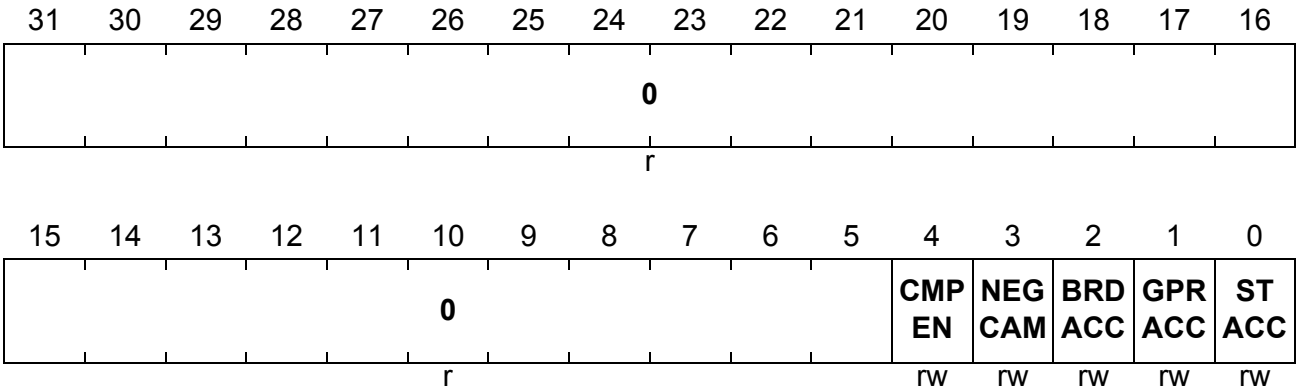
The MAC registers are used to configure and control the MAC Controller.

MACCTRL
MAC Control Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Function
HLTREQ	0	rw	**Halt Request** 0 No request to stop transmission and reception 1 Stop transmission and reception after completion of any current packets
HLTIMM	1	rw	**Halt Immediately** 0 No stop action 1 Stop transmission immediately. Stop reception after current frame is completed.
RESET	2	rw	**Reset** Reset all MAC controller state machines and FIFOs.
FULLDUP	3	rw	**Full Duplex** Allow transmission to begin while reception is occurring.
MACLP	4	rw	**MAC Loop** Cause transmission signals to be presented as input to the receive circuit without leaving the controller.
MROLL	10	r	**Missed Roll** Missed error counter rolled over.
0	[9:5], [31:11]	r	**Reserved;** read as 0; should be written with 0.

MACCAMCTRL0
MAC CAM Control Register 0 **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Function
STACC	0	rw	**Station Accept** Accept any packet with a unicast station address.
GPRACC	1	rw	**Group Access** Accept any packet with a multicast-group address.
BRDACC	2	rw	**Broadcast Access** Accept any packet with a broadcast address.
NEGCAM	3	rw	**Negative CAM** 0 Accept packets CAM recognizes, reject others 1 Reject packets CAM recognizes, accept others
CMPEN	4	rw	**Compare Enable** Enable compare mode. The CAM memory read for addresses to filter incoming messages.
0	[31:5]	r	**Reserved;** read as 0; should be written with 0.

MACTXCTRL
MAC Transmit Control Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								MII 10	SD PSE	NO DEF	0	NO CRC	NO PAD	TX HLT	TX EN
r								rw	rw	rw	r	rw	rw	rw	rw

Field	Bits	Type	Function
TXEN	0	rw	**Transmit Enable** 0 Stop transmission immediately 1 Start/continue transmission
TXHLT	1	rw	**Transmit Halt** Halt transmission immediately.
NOPAD	2	rw	**No Pad** Do not generate pad bytes for packets of less than 64 bytes.
NOCRC	3	rw	**No CRC** Do not add the CRC at the end of a packet.
NODEF	5	rw	**No Defer** Suppress the checking of excessive deferral.
SDPSE	6	rw	**Send Pause** Send a Pause command or other MAC control packet.
MII10	7	rw	**MII 10 Mbit/s Mode** Enable SQE checking in MII 10 Mbit/s mode.
0	4, [31:8]	r	**Reserved;** read as 0; should be written with 0.

MACTXSTAT
MAC Transmit Status Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															SQ ERR
r															rh

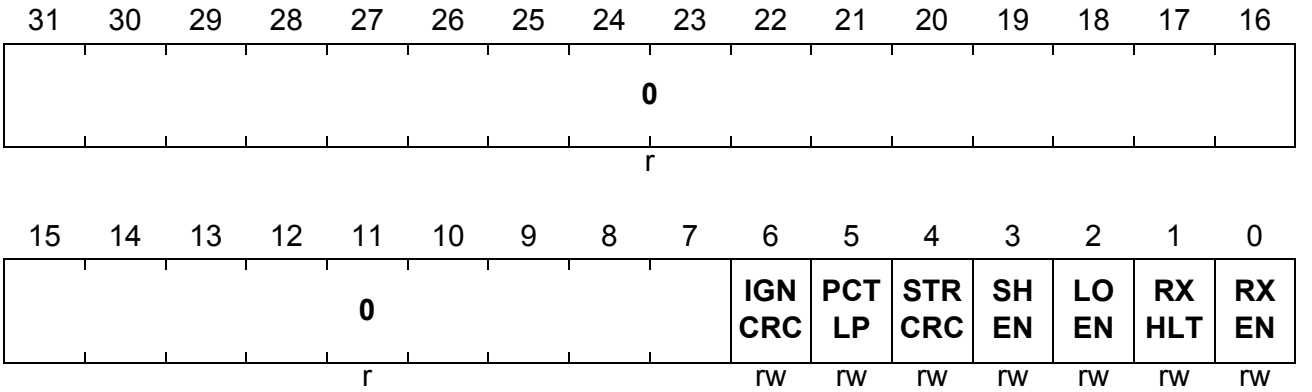
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXH LTD	CMP	TXP	L COL	0	L CAR	X DEF	UN DER	0	PSE	TX DEF	X COL	TXCOLCNT			
rh	rh	rh	rh	r	rh	rh	rh	r	rh	rh	rh	rh			

Field	Bits	Type	Description
TXCOLCNT	[3:0]	rh	**Transmit Collision Count** Count of the collisions in transmitting a single packet. If 16 collisions occur, TXCOLCNT will be zero and XCOL is set.
XCOL	4	rh	**Excessive Collision** Set if 16 collisions occur for same packet. Transmission abandoned.
TXDEF	5	rh	**Transmit Deferred** Packet waited because of deferral during transmission.
PSE	6	rh	**Paused** The following packet will be paused
UNDER	8	rh	**Underrun** MAC transmit FIFO becomes empty during transmission.
XDEF	9	rh	**Excessive Deferral** MAC defers for MAC_Derferral: = 0.24288 ms for 100 Mbit/s = 2.42880 ms for 10 Mbit/s
LCAR	10	rh	**Lost Carrier** No carrier sense detected or carrier was dropped during the transmission of a packet.
LCOL	12	rh	**Late Collision** A collision occurs after 512 bit times (64 byte times)
TXP	13	rh	**Transmit Parity** MAC transmit FIFO has detected a parity error.

Field	Bits	Type	Description
CMP	14	rh	**Completion** MAC transmits or discards one packet.
TXHLTD	15	rh	**Transmit Halted** Transmission is halted by clearing TXEN or setting TXHLT.
SQERR	16	rh	**Signal Quality Error** No heart beat signal observed at end of transmission.
0	7, 11, [31:17]	r	**Reserved;** read as 0; should be written with 0.

MACRXCTRL
MAC Receive Control Register

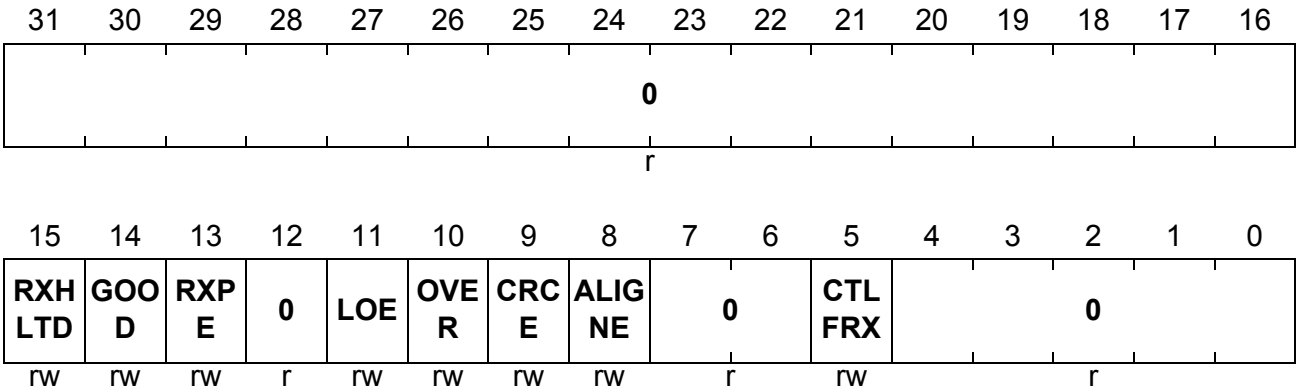
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
RXEN	0	rw	**Receive Enable** 0 Stop reception immediately
RXHLT	1	rw	**Receive Halt** Halt reception after completing any current packet.
LOEN	2	rw	**Long Enable** Allow reception of frames longer than 1518 bytes.
SHEN	3	rw	**Short Enable** Allow reception of frames shorter than 64 bytes.
STRCRC	4	rw	**Strip CRC** Check the CRC, but strip from the message.
PCTLP	5	rw	**Pass Control Packets** Enable passing of control packets to MAC client.
IGNCRC	6	rw	**Ignore CRC** Do not check the CRC.
0	[31:7]	r	**Reserved;** read as 0; should be written with 0.

MACRXSTAT
MAC Receive Status Register

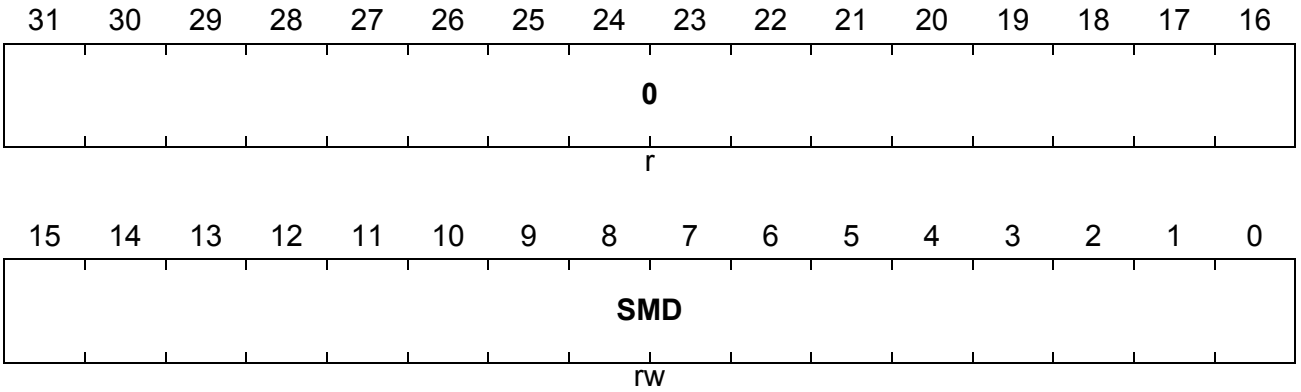
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
CTLFRX	5	rw	**Control Frame Received** Set if packet received is a MAC control frame, i.e. type = 8808_H, and CAM recognizes address.
ALIGNE	8	rw	**Alignment Error** Frame length in bits is not a multiple of eight and the CRC is invalid.
CRCE	9	rw	**CRC Error** CRC at end of packet does not match computed value, or the PHY asserted RXER during packet reception.
OVER	10	rw	**Overflow** The MAC receive FIFO is full when it need to store a received byte.
LOE	11	rw	**Long Error** Received a frame longer than 1518 bytes. Not set if the long enable but in the receive control register is set.
RXPE	13	rw	**Receive Parity Error** The MAC receive FIFO has detected a parity error.
GOOD	14	rw	**Good** Successfully received a packet with no errors.
RXHLTD	15	rw	**Receive Halted** Reception halted by clearing RXEN or setting RXHLT.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

MACSMDATA
MAC Station Management Data Register

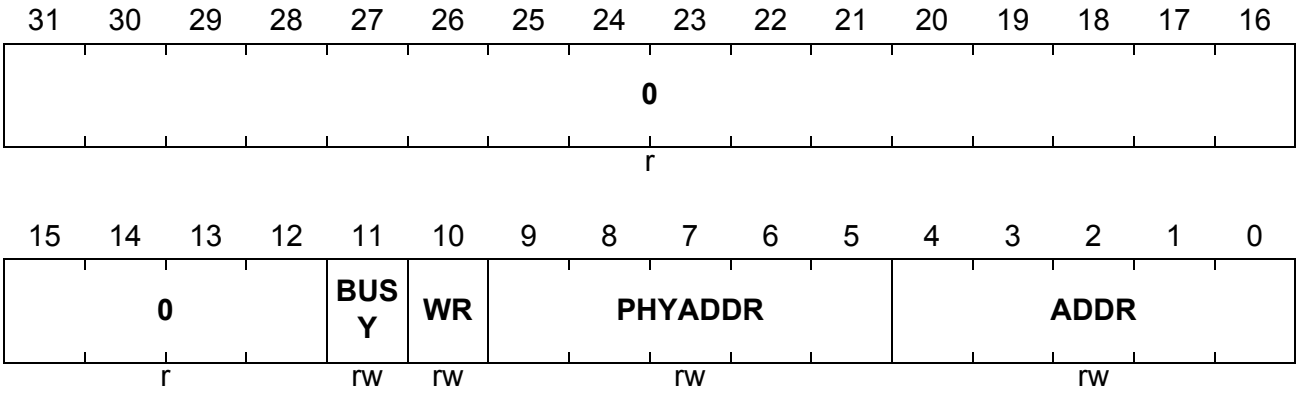
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
SMD	[15:0]	rw	**Station Management Data** Station management data written to or read from the PHY.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

MACSMCTRL
MAC Station Management Control Register

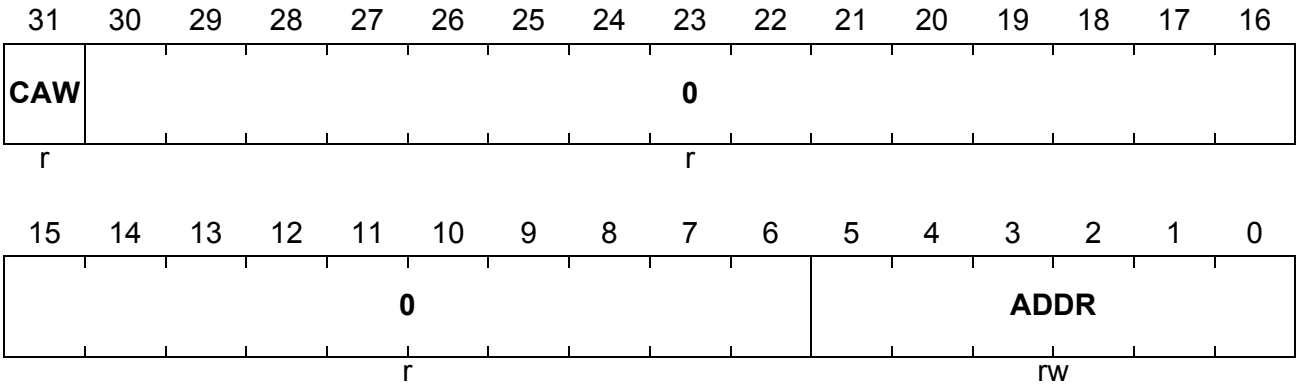
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
ADDR	[4:0]	rw	**Address** Address inside the PHY of register to read or write.
PHYADDR	[9:5]	rw	**PHY Address** Address of PHY device to read or write.
WR	10	rw	**Write/Read** 0 Read Access 1 Write Access
BUSY	11	rw	**BUSY** Set to begin operation; Controller clears when operation completes.
0	[31:12]	r	**Reserved;** read as 0; should be written with 0.

MACCAMADDR
MAC CAM Address Register

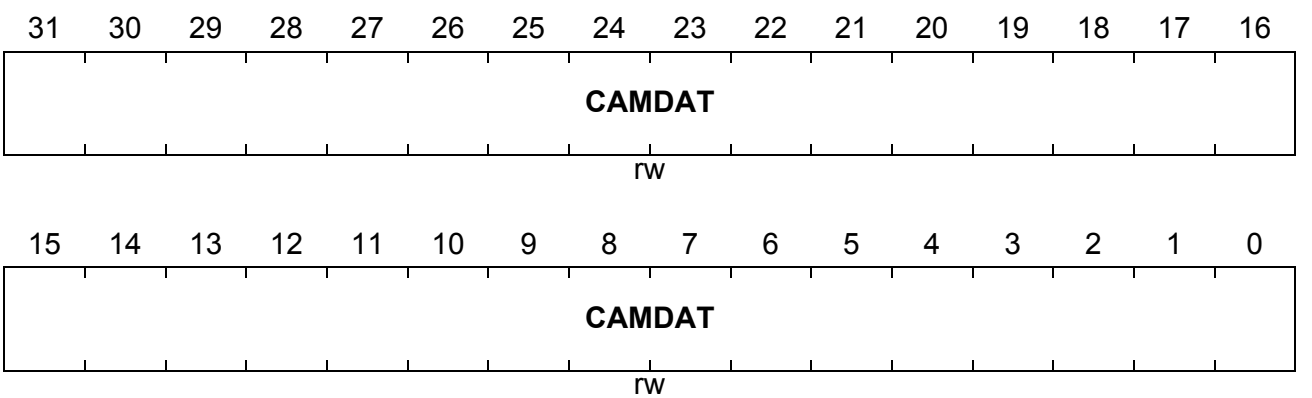
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
ADDR	[5:0]	rw	**Address** Address of location in CAM.
CAW	31	r	**CAM At Write** CAM at write indicates that currently new CAM data are written to the CAM. A new write command must not be set as long as this bit is set to 1.
0	[30:6]	r	**Reserved;** read as 0; should be written with 0.

MACCAMDATA
MAC CAM Data Register

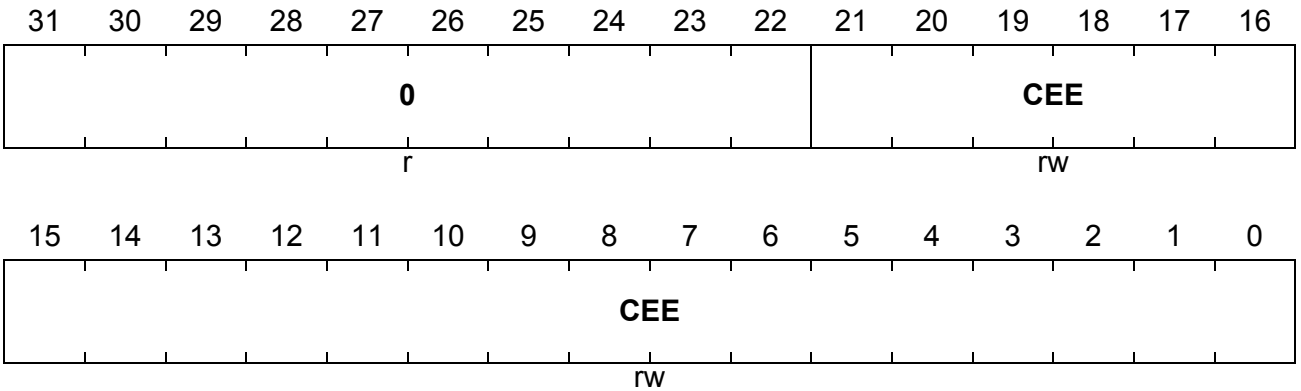
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
CAMDAT	[31:0]	rw	**CAM Data** Single CAM data word of CAM memory, which is 32 bits wide and 35 32-bit words deep.

MACCAMCTRL1
MAC CAM Control Register 1

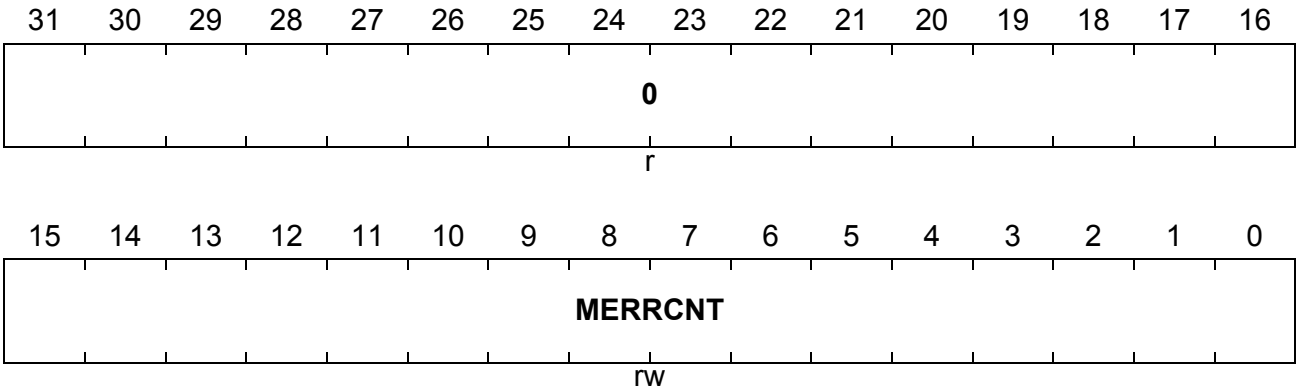
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
CEE	[21:0]	rw	**CAM Entry Enable** The CAM memory contains 35 32-bit words providing space for • up to 21 6-byte addresses • 2 reserved bytes • 3 double-words (each of 4 bytes) for Pause operation Each single address can be enabled for address comparison setting the corresponding CEE bit.
0	[30:6]	r	**Reserved;** read as 0; should be written with 0.

MACMERRCNT
MAC Missed Error Count Register

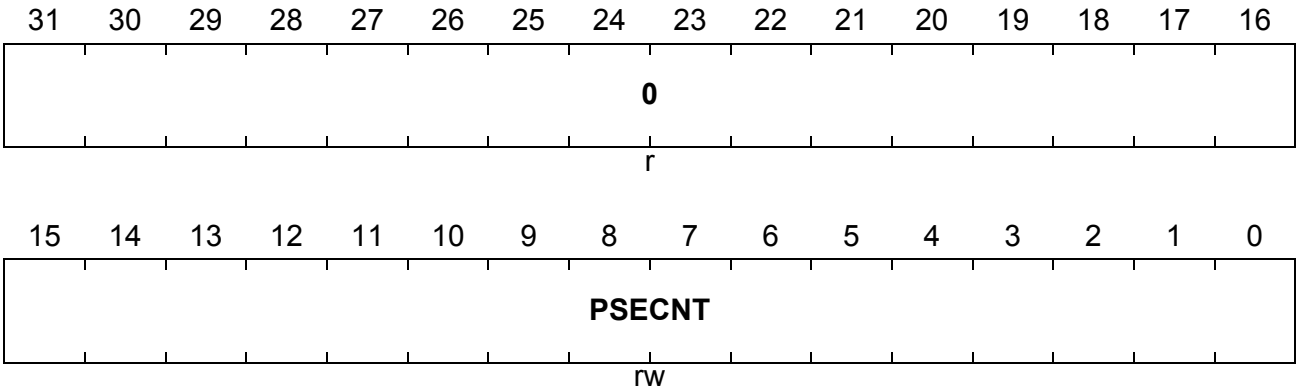
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
MERRCNT	[15:0]	rw	**Missed Error Counter** The number of valid packets which are rejected by the MAC unit because the MAC receive FIFO overflows, a parity error occurs, or the Receive Enable bit (RXEN) is cleared. This count excludes packets the CAM rejects. The Missed Error Counter rolling over from 0x7FFF to 0x8000 sets the Missed Roll bit in the MAC control register. It also generates an interrupt if enabled.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

MACPSECNT
MAC Pause Count Register

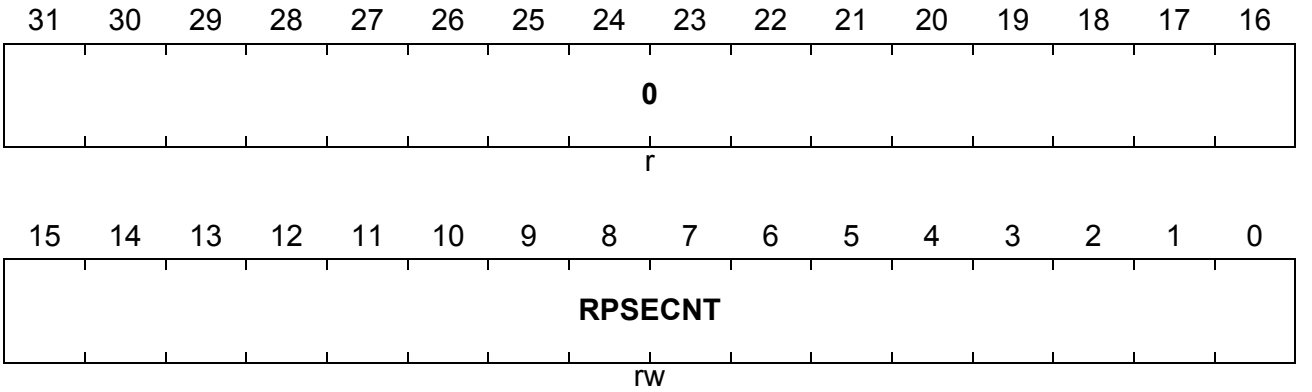
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
PSECNT	[15:0]	rw	**Receive Pause Counter** Count of time slots that transmitter is being paused, as the result of receiving a MAC control Pause operation packet.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

MACRPSECNT
MAC Remote Pause Count Register

Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
RPSECNT	[15:0]	rw	**Remote Pause Counter** Count of time slots that remote MAC is being paused, as the result of sending a Pause operation packet.
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

MACTX0IMR
MAC Transmit 0 Interrupt Mask Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ ERR	TXH LTD	CMP	TXP	L COL	L CAR	X DEF	UND ER	PSE	PAU SE	TX DEF	X COL	0			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r			

Field	Bits	Type	Description
XCOL	4	rw	**Excessive Collision** 0 Disable the Excessive Collision Interrupt 1 Enable the Excessive Collision Interrupt
TXDEF	5	rw	**Transmit Deferred** 0 Disable the Transmit Deferred Interrupt 1 Enable the Transmit Deferred Interrupt
PAUSE	6	rw	**Pause** 0 Disable the External Request for the Pause Interrupt 1 Enable the External Request for the Pause Interrupt
PSE	7	rw	**Pause Frame Sent** 0 Disable MAC Sending a Pause Frame Interrupt 1 Enable MAC Sending a Pause Frame Interrupt
UNDER	8	rw	**Underrun** 0 Disable the Underrun Interrupt 1 Enable the Underrun Interrupt
XDEF	9	rw	**Excessive Deferral** 0 Disable the excessive Deferral Interrupt 1 Enable the excessive Deferral Interrupt
LCAR	10	rw	**Lost Carrier** 0 Disable the Lost Carrier Interrupt 1 Enable the Lost carrier Interrupt
LCOL	11	rw	**Late Collision** 0 Disable the Late Collision Interrupt 1 Enable the Late Collision Interrupt

Field	Bits	Type	Description
TXP	12	rw	**Transmit Parity** 0 Disable the Transmit Parity Interrupt 1 Enable the Transmit Parity Interrupt
CMP	13	rw	**Completion** 0 Disable the Completion Interrupt 1 Enable the Completion Interrupt
TXHLTD	14	rw	**Transmit Halted** 0 Disable the Transmit Halted Interrupt 1 Enable the Transmit Halted Interrupt
SQERR	15	rw	**Signal Quality Error** 0 Disable the Signal Quality Error Interrupt 1 Enable the Signal Quality Error Interrupt
0	[3:0], [30:16]	r	**Reserved;** read as 0; should be written with 0.

MACTX0ISR
MAC Transmit 0 Interrupt Status Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ ERR	TXH LTD	CMP	TXP	L COL	L CAR	X DEF	UND ER	PSE	PAU SE	TX DEF	X COL	TXCOLCNT			
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh			

Field	Bits	Type	Description
TXCOLCNT	[3:0]	rh	**Transmit Collision Count** Count of the collisions in transmitting a single packet, if 16 collisions occur, TXCOLCNT will be zero and XCOL is set.
XCOL	4	rh	**Excessive Collision** Set if 16 collisions occur for same packet. Transmission abandoned.
TXDEF	5	rh	**Transmit Deferred** Packet waited because of deferral during transmission.
PAUSE	6	rh	**Pause** Set if MAC receives an external request for the pause.
PSE	7	rh	**Pause Frame Sent** Set if the MAC sends a Pause frame.
UNDER	8	rh	**Underrun** MAC transmit FIFO becomes empty during transmission.
XDEF	9	rh	**Excessive Deferral** MAC deferred for MAX_deferral: = 0.24288 ms for 100 Mbit/s = 2.42880 ms for 10 Mbit/s.
LCAR	10	rh	**Lost Carrier** No carrier sense detected or carrier is dropped during the transmission of a packet.
LCOL	11	rh	**Late Collision** A collision occurs after 512 bit times (64 byte times).

Field	Bits	Type	Description
TXP	12	rh	**Transmit Parity** MAC transmit FIFO has detected a parity error.
CMP	13	rh	**Completion** MAC transmission or discards one packet.
TXHLTD	14	rh	**Transmit Halted** Transmission is halted by clearing TXEN or setting TXHLT.
SQERR	15	rh	**Signal Quality Error** No heart beat signal observed at end of transmission.
0	[30:16]	r	**Reserved;** read as 0; should be written with 0.

MACTX1IMR
MAC Transmit 1 Interrupt Mask Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ ERR	TXH LTD	CMP	TXP	L COL	L CAR	X DEF	UND ER	PSE	PAU SE	TX DEF	X COL	0			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r			

Field	Bits	Type	Description
XCOL	4	rw	**Excessive Collision** 0 Disable the Excessive Collision Interrupt 1 Enable the Excessive Collision Interrupt
TXDEF	5	rw	**Transmit Deferred** 0 Disable the Transmit Deferred Interrupt 1 Enable the Transmit Deferred Interrupt
PAUSE	6	rw	**Pause** 0 Disable the External Request for the Pause Interrupt 1 Enable the External Request for the Pause Interrupt
PSE	7	rw	**Pause Frame Sent** 0 Disable MAC Sending a Pause Frame Interrupt 1 Enable MAC Sending a Pause Frame Interrupt
UNDER	8	rw	**Underrun** 0 Disable the Underrun Interrupt 1 Enable the Underrun Interrupt
XDEF	9	rw	**Excessive Deferral** 0 Disable the Excessive Deferral Interrupt 1 Enable the Excessive Deferral Interrupt
LCAR	10	rw	**Lost Carrier** 0 Disable the Lost Carrier Interrupt 1 Enable the Lost Carrier Interrupt
LCOL	11	rw	**Late Collision** 0 Disable the Late Collision Interrupt 1 Enable the Late Collision Interrupt

Field	Bits	Type	Description
TXP	12	rw	**Transmit Parity** 0 Disable the Transmit Parity Interrupt 1 Enable the Transmit Parity Interrupt
CMP	13	rw	**Completion** 0 Disable the Completion Interrupt 1 Enable the Completion Interrupt
TXHLTD	14	rw	**Transmit Halted** 0 Disable the Transmit Halted Interrupt 1 Enable the Transmit Halted Interrupt
SQERR	15	rw	**Signal Quality Error** 0 Disable the Signal Quality Error Interrupt 1 Enable the Signal Quality Error Interrupt
0	[3:0], [30:16]	r	**Reserved;** read as 0; should be written with 0.

MACTX1ISR
MAC Transmit 1 Interrupt Status Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

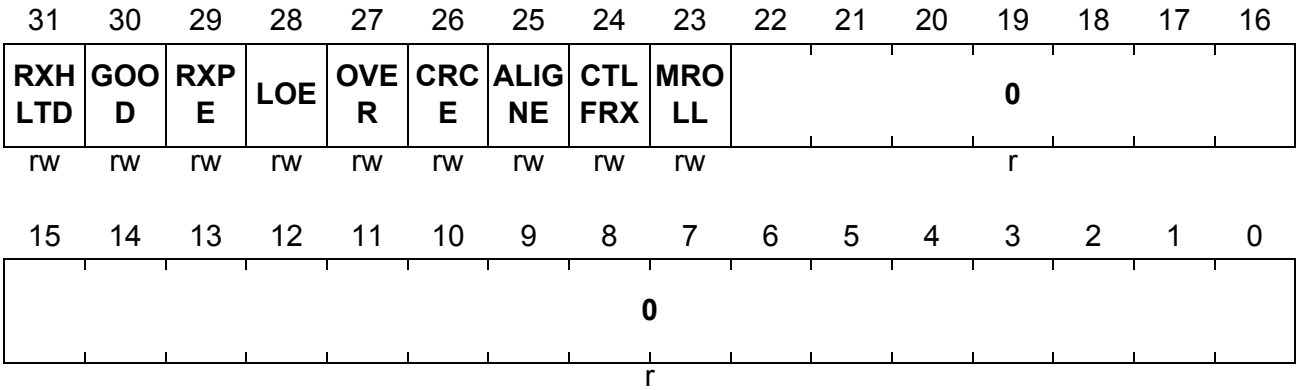
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ ERR	TXH LTD	CMP	TXP	L COL	L CAR	X DEF	UND ER	PSE	PAU SE	TX DEF	X COL	TXCOLCNT			
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh			

Field	Bits	Type	Description
TXCOLCNT	[3:0]	rh	**Transmit Collision Count** Count of the collisions in transmitting a single packet, if 16 collisions occur, TXCOLCNT will be zero and XCOL is set.
XCOL	4	rh	**Excessive Collision** Set if 16 collisions occur for same packet. Transmission abandoned.
TXDEF	5	rh	**Transmit Deferred** Packet waited because of deferral during transmission.
PAUSE	6	rh	**Pause** Set if MAC receives an external request for the pause.
PSE	7	rh	**Pause Frame Sent** Set if the MAC sends a Pause frame.
UNDER	8	rh	**Underrun** MAC transmit FIFO becomes empty during transmission.
XDEF	9	rh	**Excessive Deferral** MAC deferred for MAX_deferral: = 0.24288 ms for 100 Mbit/s = 2.42880 ms for 10 Mbit/s.
LCAR	10	rh	**Lost Carrier** No carrier sense detected or carrier is dropped during the transmission of a packet.
LCOL	11	rh	**Late Collision** A collision occurs after 512 bit times (64 byte times).

Field	Bits	Type	Description
TXP	12	rh	**Transmit Parity** MAC transmit FIFO has detected a parity error.
CMP	13	rh	**Completion** MAC transmission or discards one packet.
TXHLTD	14	rh	**Transmit Halted** Transmission is halted by clearing TXEN or setting TXHLT.
SQERR	15	rh	**Signal Quality Error** No heart beat signal observed at end of transmission.
0	[30:16]	r	**Reserved;** read as 0; should be written with 0.

MACRX0ISR
MAC Receive 0 Interrupt Status Register

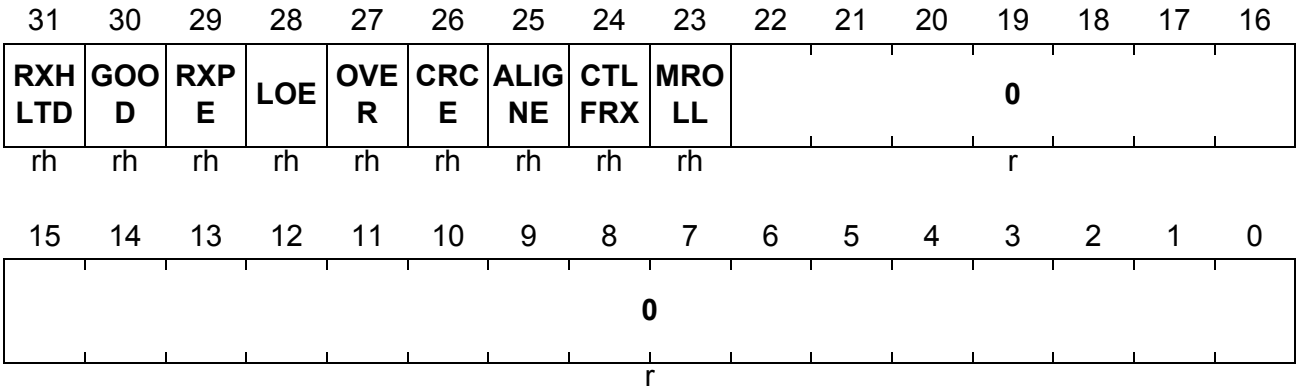
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
MROLL	23	rw	**Missed Error Counter Roll** Missed error counter rolled over indication.
CTLFRX	24	rw	**Control Frame Received** Set if packet received is a MAC control frame, i.e. type = 8808_H, and CAM recognizes address.
ALIGNE	25	rw	**Alignment Error** Frame length in bits is not a multiple of eight and the CRC is invalid.
CRCE	26	rw	**CRC Error** CRC at end of packet does not match computed value, or the PHY asserted RXER during packet reception.
OVER	27	rw	**Overflow** The MAC receive FIFO is full when it need to store a received byte.
LOE	28	rw	**Long Error** Received a frame longer than 1518 bytes. Not set if the long enable bit in the receive control register is set.
RXPE	29	rw	**Receive Parity Error** The MAC receive FIFO has detected a parity error.
GOOD	30	rw	**Good** Successfully received a packet with no errors.

Field	Bits	Type	Description
RXHLTD	31	rw	**Receive Halted** Reception halted by clearing RXEN or setting RXHLT. After initialization of MAC receive, this interrupt must be cleared by software and ignored.
0	[22:0]	r	**Reserved;** read as 0; should be written with 0.

MACRX0IMR
MAC Receive 0 Interrupt Mask Register **Reset Value: 0000 0000$_H$**

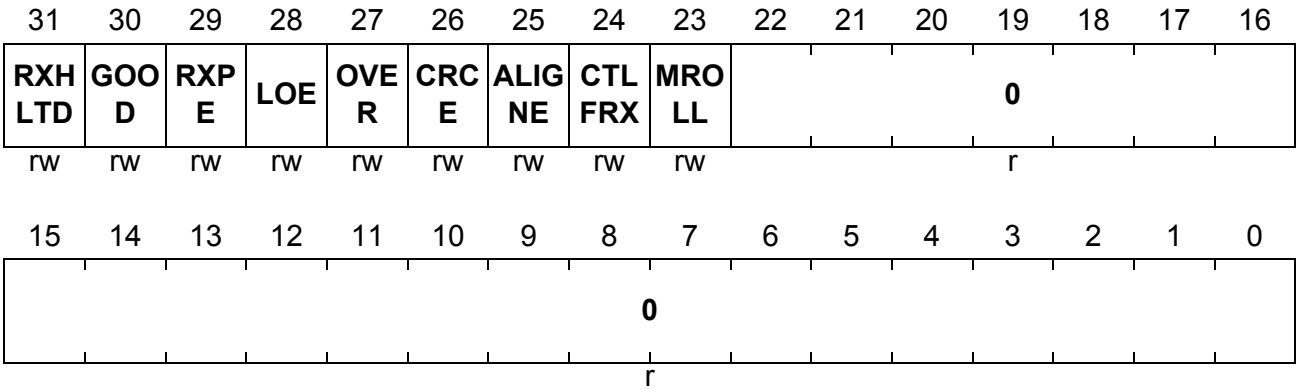


Field	Bits	Type	Description
MROLL	23	rh	**Missed Error Counter Roll** 0 Disable the Missed Error Counter Roll Interrupt 1 Enable the Missed Error Counter Roll Interrupt
CTLFRX	24	rh	**Control Frame Received** 0 Disable the Control Frame Received Interrupt 1 Enable the Control Frame Received Interrupt
ALIGNE	25	rh	**Alignment Error** 0 Disable the Alignment Error Interrupt 1 Enable the Alignment Error Interrupt
CRCE	26	rh	**CRC Error** 0 Disable the CRC Error Interrupt 1 Enable the CRC Error Interrupt
OVER	27	rh	**Overflow** 0 Disable the Overflow Interrupt 1 Enable the Overflow Interrupt
LOE	28	rh	**Long Error** 0 Disable the Long Error Interrupt 1 Enable the Long Error Interrupt
RXPE	29	rh	**Receive Parity Error** 0 Disable the Receive parity Error Interrupt 1 Enable the Receive parity Error Interrupt
GOOD	30	rh	**Good** 0 Disable the Good Interrupt 1 Enable the Good Interrupt

Field	Bits	Type	Description
RXHLTD	31	rh	**Receive Halted** 0 Disable the Receive Halted Interrupt 1 Enable the Receive Halted Interrupt
0	[22:0]	r	**Reserved;** read as 0; should be written with 0.

MACRX1ISR
MAC Receive 1 Interrupt Status Register

Reset Value: 0000 0000$_H$

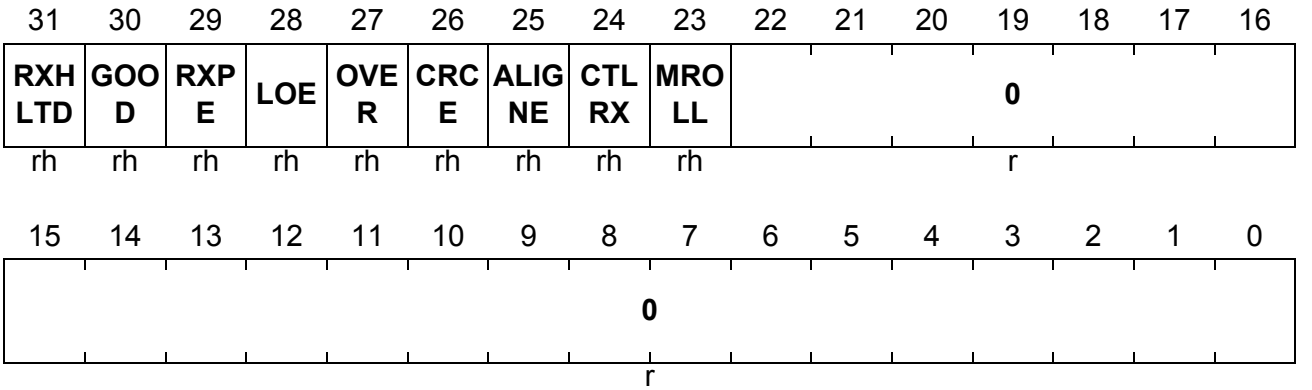


Field	Bits	Type	Description
MROLL	23	rw	**Missed Error Counter Roll** Missed error counter rolled over indication.
CTLFRX	24	rw	**Control Frame Received** Set if packet received is a MAC control frame, i.e. type = 8808_H, and CAM recognizes address.
ALIGNE	25	rw	**Alignment Error** Frame length in bits is not a multiple of eight and the CRC is invalid.
CRCE	26	rw	**CRC Error** CRC at end of packet does not match computed value, or the PHY asserted RXER during packet reception.
OVER	27	rw	**Overflow** The MAC receive FIFO is full when it need to store a received byte.
LOE	28	rw	**Long Error** Received a frame longer than 1518 bytes. Not set if the long enable bit in the receive control register is set.
RXPE	29	rw	**Receive Parity Error** The MAC receive FIFO has detected a parity error.
GOOD	30	rw	**Good** Successfully received a packet with no errors.

Field	Bits	Type	Description
RXHLTD	31	rw	**Receive Halted** Reception halted by clearing RXEN or setting RXHLT. After initialization of MAC receive, this interrupt must be cleared by software and ignored.
0	[22:0]	r	**Reserved;** read as 0; should be written with 0.

MACRX1IMR
MAC Receive 1 Interrupt Mask Register

Reset Value: 0000 0000$_H$



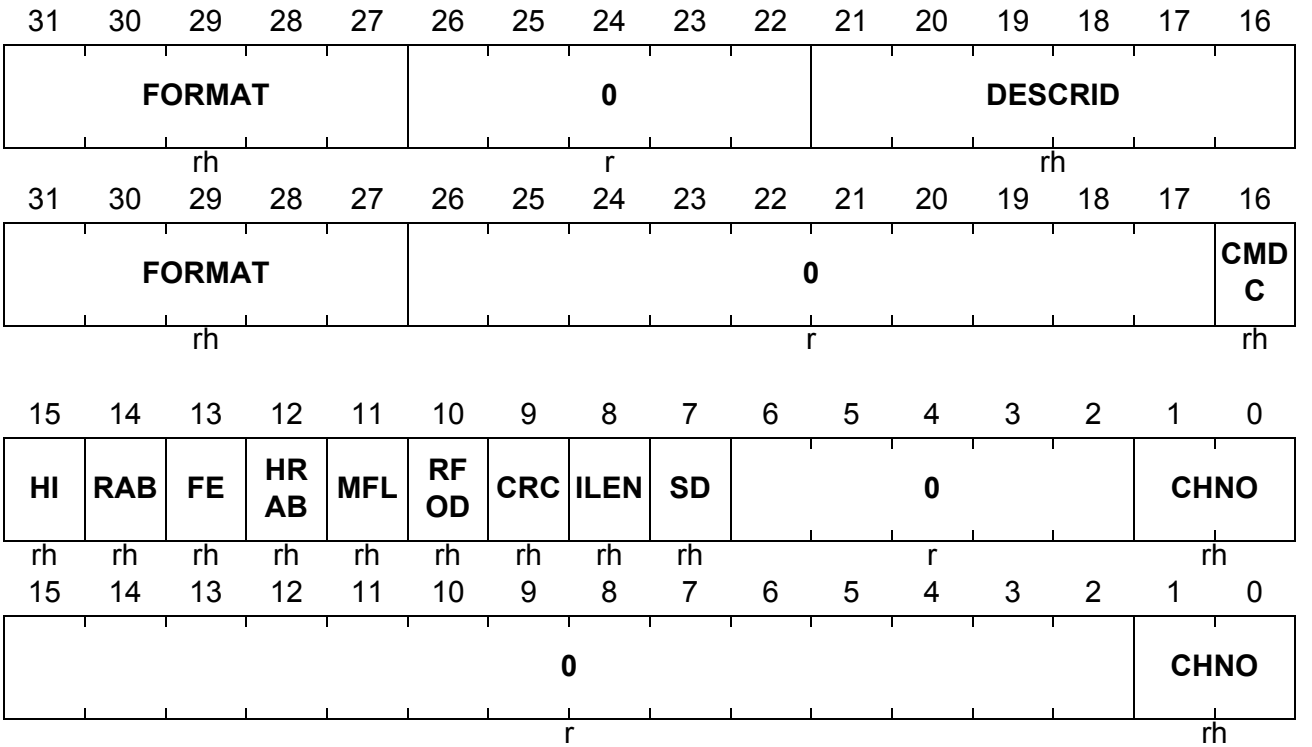
Field	Bits	Type	Description
MROLL	23	rh	**Missed Error Counter Roll** 0 Disable the Missed Error Counter Roll Interrupt 1 Enable the Missed Error Counter Roll Interrupt
CTLFRX	24	rh	**Control Frame Received** 0 Disable the Control Frame Received Interrupt 1 Enable the Control Frame Received Interrupt
ALIGNE	25	rh	**Alignment Error** 0 Disable the Alignment Error Interrupt 1 Enable the Alignment Error Interrupt
CRCE	26	rh	**CRC Error** 0 Disable the CRC Error Interrupt 1 Enable the CRC Error Interrupt
OVER	27	rh	**Overflow** 0 Disable the Overflow Interrupt 1 Enable the Overflow Interrupt
LOE	28	rh	**Long Error** 0 Disable the Long Error Interrupt 1 Enable the Long Error Interrupt
RXPE	29	rh	**Receive Parity Error** 0 Disable the Receive Parity Error Interrupt 1 Enable the Receive Parity Error Interrupt
GOOD	30	rh	**Good** 0 Disable the Good Interrupt 1 Enable the Good Interrupt

Field	Bits	Type	Description
RXHLTD	31	rh	**Receive Halted** 0 Disable the Received Halted Interrupt 1 Enable the Received Halted Interrupt
0	[22:0]	r	**Reserved;** read as 0; should be written with 0.

31.2.2 DMUR Registers

The DMUR (DR) registers are used to configure and control the DR block.

DRISFIFO
DR Interrupt Status FIFO **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CHNO	[1:0]	rh	**Channel Number** Identifies the channel number of the channel that has generated the interrupt.
SD	7	rh	**Silent Discard** A completely received frame has been discarded silently due to a HRAB condition. Reserved when register work as Command Interrupt Vector specified by FORMAT bits.
ILEN	8	rh	**Invalid Length** Indicates a frame length not multiple of 8 bits. Reserved when register work as Command Interrupt Vector specified by FORMAT bits.

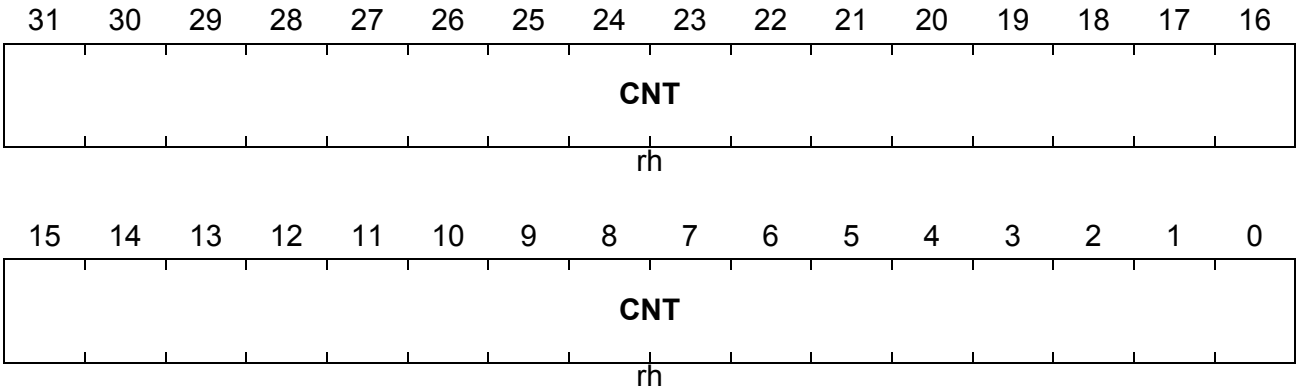
Field	Bits	Type	Description
CRC	9	rh	**CRC Error** Cyclic Redundancy Check Error. Reserved when register work as Command Interrupt Vector specified by FORMAT bits.
RFOD	10	rh	**Receive Frame Overflow** Indicates a Receive Frame Overflow. A protocol handler is unable to transfer data to the Receive Buffer (RB). As soon as RB can store data again, a RFOD status is appended to the frame. Reserved when register work as Command Interrupt Vector specified by FORMAT bits.
MFL	11	rh	**Maximum Frame Length** The packet exceeds the maximum allowed frame size. Reserved when register work as Command Interrupt Vector specified by FORMAT bits.
HRAB	12	rh	**Hold Caused Receive Abort** Issued if a frame could not be transferred to the shared memory completely because of a HOLD bit set in a receive descriptor not providing enough bytes for the frame or the frame was aborted by a Receive Abort Command. Reserved when register work as Command Interrupt Vector specified by FORMAT bits.
FE	13	rh	**Frame End** Set to 1 indicates that a frame (valid or invalid) has been received completely or is stopped by the DMUR reset or a HOLD in a receive descriptor. It is set when the DMUR branches from the last descriptor belonging to the frame to the first descriptor of a new frame. It is also set when the descriptor in which the frame finished contained a hold bit. Reserved when register work as Command Interrupt Vector specified by FORMAT bits.

Field	Bits	Type	Description
RAB	14	rh	**Receive Abort** Set when the CPU programmed a receive abort or a receive off command and therefore a frame could not be received completely. Additionally, it can be set in the status word coming from the protocol machine indicating that the frame is aborted from the serial side. Reserved when register work as Command Interrupt Vector specified by FORMAT bits.
HI	15	rh	**Host Initiated** If the HI bit in the receive descriptor is set, this bit will be activated when the DMUR finishes the current receive descriptor and branches to the next receive descriptor. This interrupt is not maskable in the Interrupt Mask register, but in the descriptor. Reserved when register operates as Command Interrupt Vector specified by FORMAT bits.
CDMC	16	rh	**Command Completed** Reserved when register work as Channel Interrupt Vector specified by FORMAT bits. Identifies a successful channel command performed by the host CPU. A new command can be given by the CPU for this channel.
DESCRID	[21:16]	rh	**Descriptor ID** Identifies the receive descriptor that generated the interrupt. Reserved except bit 16 when register operates as Command Interrupt Vector specified by FORMAT bits.
FORMAT	[31:27]	rh	**Format** This pattern specifies the format of the interrupt vector. 10101 Channel Interrupt Vector 10000 Command Interrupt Vector
0	[6:2], [26:22]	r	**Reserved;** read as 0; should be written with 0.

Note: This interrupt status FIFO contains up to 16 interrupt vectors. There are two different interrupt status words, which are differentiated by the FORMAT bits. All bits are reset when the register is read. DR does not continue with operation as long as interrupts need to be serviced.

DRFFCR
DR FIFO Counter Register

Reset Value: 0000 0000$_H$

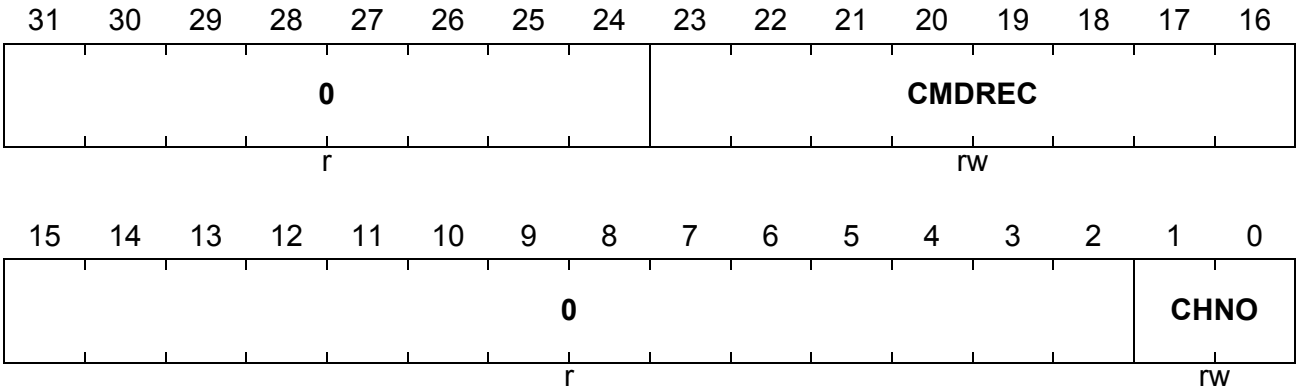


Field	Bits	Type	Description
CNT	[31:0]	rh	**Counter** This counter is incremented when an interrupt occurs and the interrupt status cannot be stored in the interrupt status FIFO due to a FIFO full condition. The DR is halted until at least one interrupt status is read from the interrupt status FIFO. In that case, the pending interrupt status is forwarded to the interrupt status FIFO and DR resumes operation. Thus, this counter indicates how often an interrupt status could not be stored due to a FIFO full condition.

Note: After the software read the value of CNT bit field, the hardware will clear it.

DRCMD
DR Command Register

Reset Value: 0000 0000$_H$

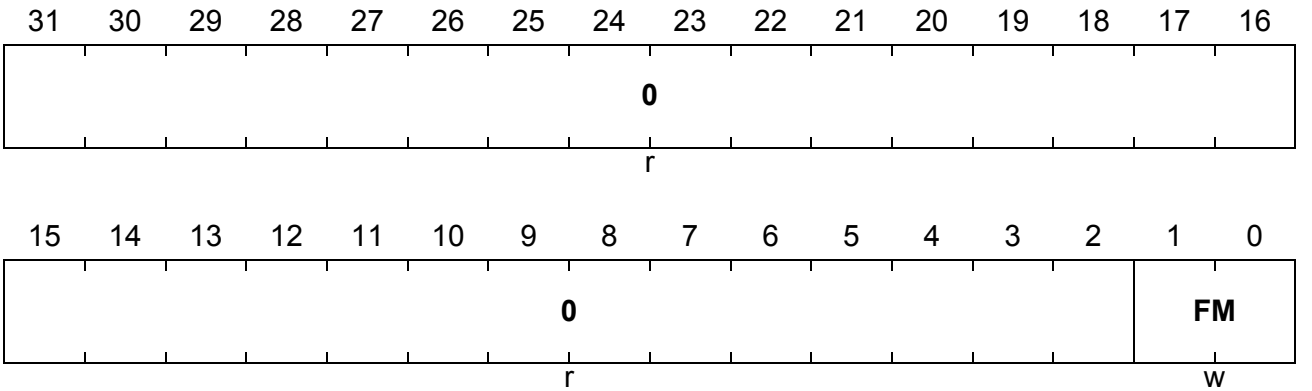


Field	Bits	Type	Description
CHNO	[1:0]	rw	**Channel Number** Identifies the channel number of the channel that has generated the interrupt; only CHNO = ‘00’ is supported.

Field	Bits	Type	Description
CMDREC	[23:16]	rw	**Command Receive** 00000000 Receive NOP 00000001 Receive Init This command will cause the initialization of the particular channel CHNO(1:0) with the parameters that have been set in the registers DRMOD, DRBUF, DRFRDA and DRCONF. The DMUR will store the First Receive Descriptor Address (register DRFRDA) in its on-chip memory. Afterwards, the DMUR will wait for the first request for this channel from the receive buffer. Based on this request, the DMUR will fetch the entire receive descriptor pointed by the DRFRDA register. When this task is completed, a command completed interrupt CDMC is generated. (Only CHNO = ‘00’ is supported). 00000010 Receive Off Setting this bit will cause the DMUR to turn the channel off. As long as the DMUR does not see the Receive Off status word, it works normally. If the status word is seen by the DMUR and a frame has not been transferred completely, it will generate a Receive Abort Interrupt. The current receive descriptor will be released setting the C, FE and write the abort status. Afterwards or when the last frame was transferred completely a Command Complete Interrupt will be generated. The next command for this channel must be a Receive Init. The DMUR does not expect any data for the channel after the receive off status word was ready by the macro until the next initialization. 00000100 Receive Abort If a receive abort is detected, the current receive descriptor will be suspended. the DMUR stores the FRDA in its on-chip RAM. 00001000 Reserved 00010000 Receive Hold Reset After the CPU removes the HOLD bit in the receive descriptor chain of the channel, it is recommended to write the Receive Hold Reset Command.
0	[15:2], [31:24]	r	**Reserved;** read as 0; should be written with 0.

DRMOD
DR Mode Register

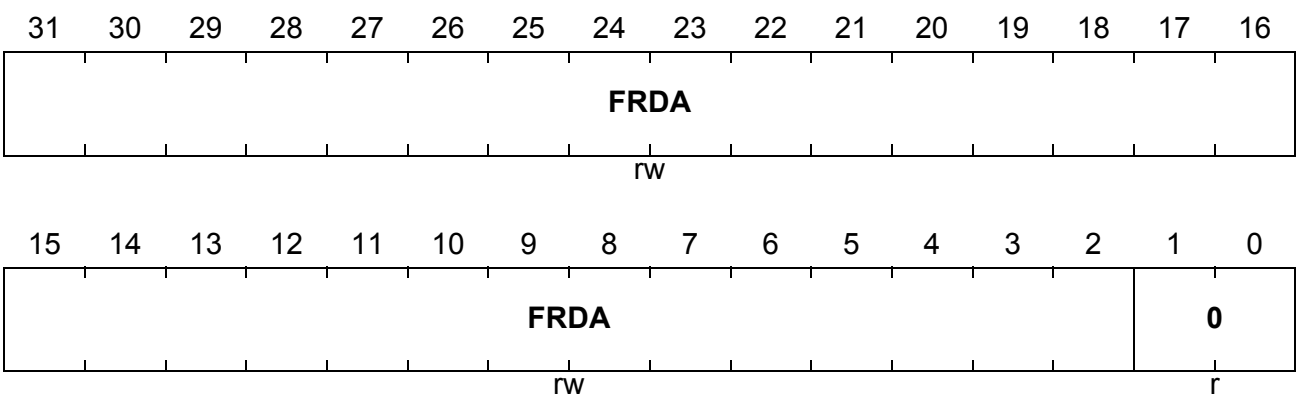
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
FM	[1:0]	w	**Frame Mode** This field must always be set to ‘00’. 00 Select Frame Mode (Ethernet) 01 Reserved 10 Reserved 11 Reserved
0	[31:2]	r	**Reserved;** read as 0; should be written with 0.

DRFRDA
DR First Rx Descriptor Address Register

Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
FRDA	[31:2]	rw	**First Receive Descriptor Address** Address of the first receive descriptor.
0	[1:0]	r	**Reserved;** read as 0; should be written with 0.

DRIMR
DR Interrupt Mask Register

Reset Value: 0000 7F05$_H$

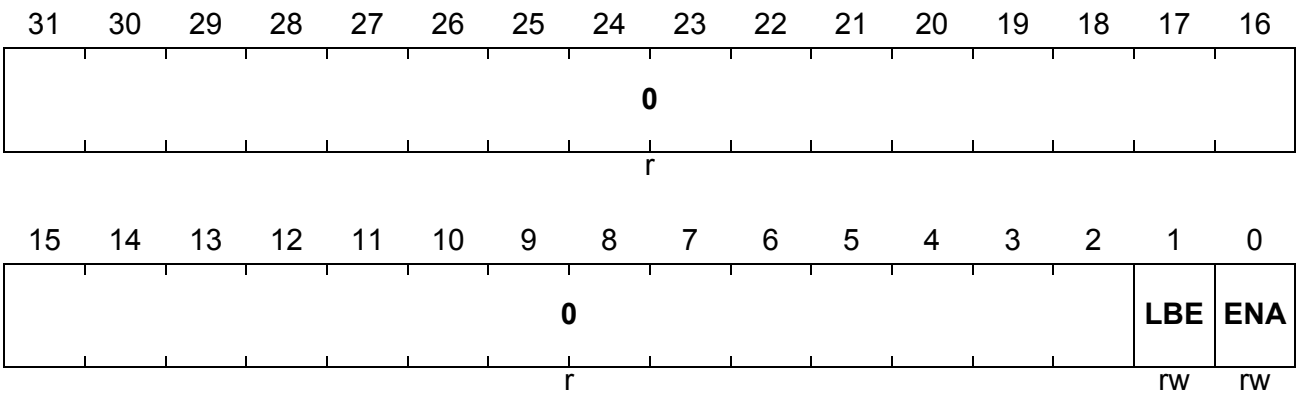
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	RAB	FE	HRAB	MFL	RFOD	CRC	ILEN	0					SD	0	CMDC
rw	rw	rw	rw	rw	rw	rw	rw	r					rw	r	rw

Field	Bits	Type	Description
CMDC	0	rw	**Command Completed Interrupt Mask** 0 Interrupt generation is enabled 1 Interrupt generation is disabled
SD	2	rw	**Silent Discard Interrupt Mask** 0 Interrupt generation is enabled 1 Interrupt generation is disabled
ILEN	8	rw	**Invalid Length Interrupt Mask** 0 Interrupt generation is enabled 1 Interrupt generation is disabled
CRC	9	rw	**CRC Error Interrupt Mask** 0 Interrupt generation is enabled 1 Interrupt generation is disabled
RFOD	10	rw	**Receive Frame Overflow Interrupt Mask** 0 Interrupt generation is enabled 1 Interrupt generation is disabled
MFL	11	rw	**Maximum Frame Length Interrupt Mask** 0 Interrupt generation is enabled 1 Interrupt generation is disabled
HRAB	12	rw	**Hold Caused Receive Abort Interrupt Mask** 0 Interrupt generation is enabled 1 Interrupt generation is disabled
FE	13	rw	**Frame End Interrupt Mask** 0 Interrupt generation is enabled 1 Interrupt generation is disabled

Field	Bits	Type	Description
RAB	14	rw	**Receive Abort Interrupt Mask** 0 Interrupt generation is enabled 1 Interrupt generation is disabled
0	1, [7:3], [31:15]	r	**Reserved;** read as 0; should be written with 0.

DRCONF
DR Configuration Register **Reset Value: 0000 0001$_H$**

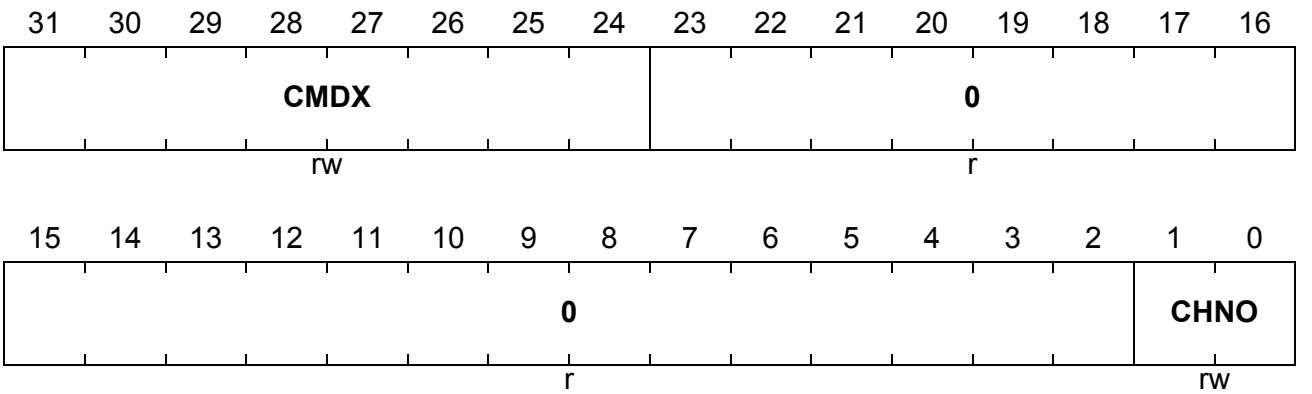


Field	Bits	Type	Description
ENA	0	rw	**Enable Alignment** 0 Block boundary alignment for bursts is disabled 1 Block boundary alignment for bursts is enabled
LBE	1	rw	**Little/Big Endian Byte Swap** 0 Little Endian 1 Big Endian
0	[31:2]	r	**Reserved;** read as 0; should be written with 0.

31.2.3 DMUT Registers

The DMUT (DT) registers are used to configure and control the DT block.

DTCMD
DT Command Register **Reset Value: 0000 0000$_H$**



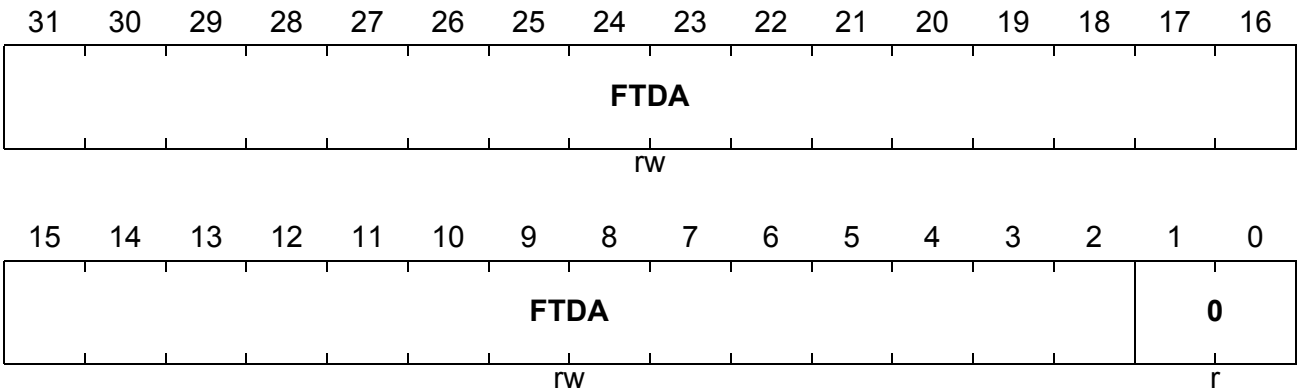
Field	Bits	Type	Description
CHNO	[1:0]	rw	**Channel Number** Identifies the channel number of the channel for which the command is issued; only CHNO = ‘00’ is supported.

Field	Bits	Type	Description
CMDX	[31:24]	rw	**Command Transmit** 00000000 Transmit NOP 00000001 Transmit Init This command will cause the initialization of the particular channel CHNO(1:0) with the parameters that have been set in the registers DTMOD, DTFTDA, and DTCONF. The DMUT will store the First Transmit Descriptor Address (register DTFTDA) in its on-chip memory. With the next request for this channel from the Transmit Buffer, the entire transmit descriptor pointed by the DTFTDA will be retrieved and afterwards a Command Complete will be generated. (Only CHNO = '00' is supported). 00000010 Transmit Off Setting this bit will cause the DMUT to turn the channel off. The DMUT updates the last processed descriptor with the complete bit if enabled and generates a 'Host Initiated' interrupt if HI bit in the last descriptor is set. All channel related information is cleared from the internal channel database. 00000100 Transmit Abort/Branch The DMUT stops immediately processing the current descriptor and branches to a new descriptor pointed at by DTFTDA. Data already stored in the Transmit Buffer is sent on the serial side. The protocol machine will append an abort sequence if data in Transmit Buffer is not completed due to 'Transmit Abort/ Branch' command. System software is informed about the aborted frame by a 'Transmit Abort' channel interrupt. 00001000 Transmit Hold Reset The 'Transmit Hold Reset' command must be given after system software has set the HOLD bit of a descriptor from 1 to 0. In case the DMUT is in hold condition, it reads the descriptor which had its HOLD bit set and tests the HOLD bit of the descriptor. 00010000 Reserved

Field	Bits	Type	Description
			If the HOLD bit is set to 0, the DMUT branches to the next descriptor and continues data transmission. Otherwise, the particular channel remains in hold condition. The DMUT will not generate a 'Transmit Command Complete' interrupt vector after this command is programmed.
0	[23:2]	r	**Reserved;** read as 0; should be written with 0.

DTFTDA
DT First Tx Descriptor Address Register

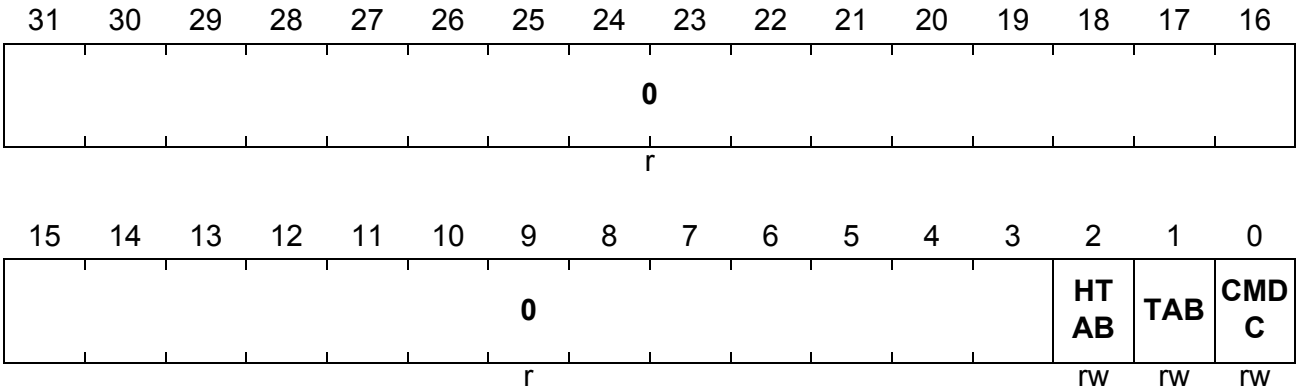
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
FTDA	[31:2]	rw	**First Transmit Descriptor Address** Address of the first transmit descriptor.
0	[1:0]	r	**Reserved;** read as 0; should be written with 0.

DTIMR
DT Interrupt Mask Register

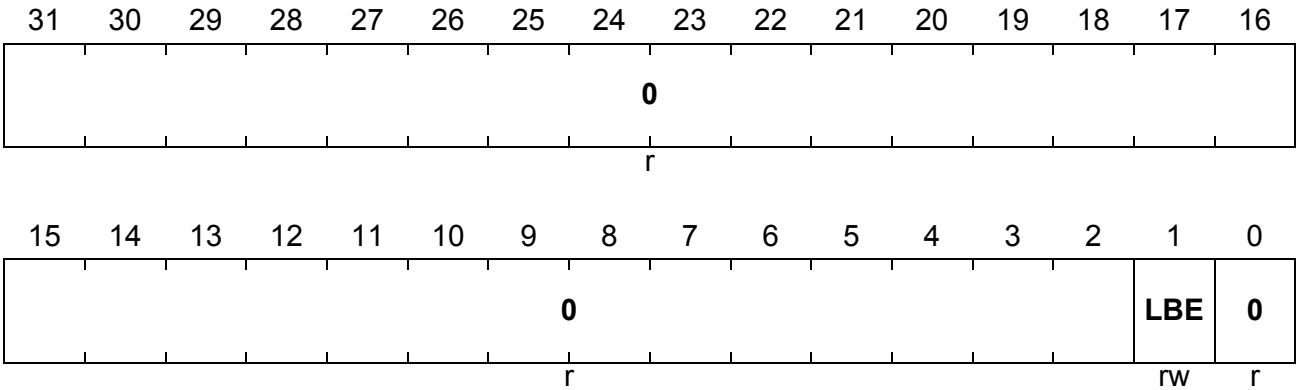
Reset Value: 0000 0007$_H$



Field	Bits	Type	Description
CMDC	0	rw	**Command Completed Interrupt Mask** 0 Interrupt generation is enabled 1 Interrupt generation is disabled
TAB	1	rw	**Transmit Abort Interrupt Mask** 0 Interrupt generation is enabled 1 Interrupt generation is disabled
HTAB	2	rw	**Hold Caused Transmit Abort Interrupt Mask** 0 Interrupt generation is enabled 1 Interrupt generation is disabled
0	[31:3]	r	**Reserved;** read as 0; should be written with 0.

DTCONF
DT Configuration Register

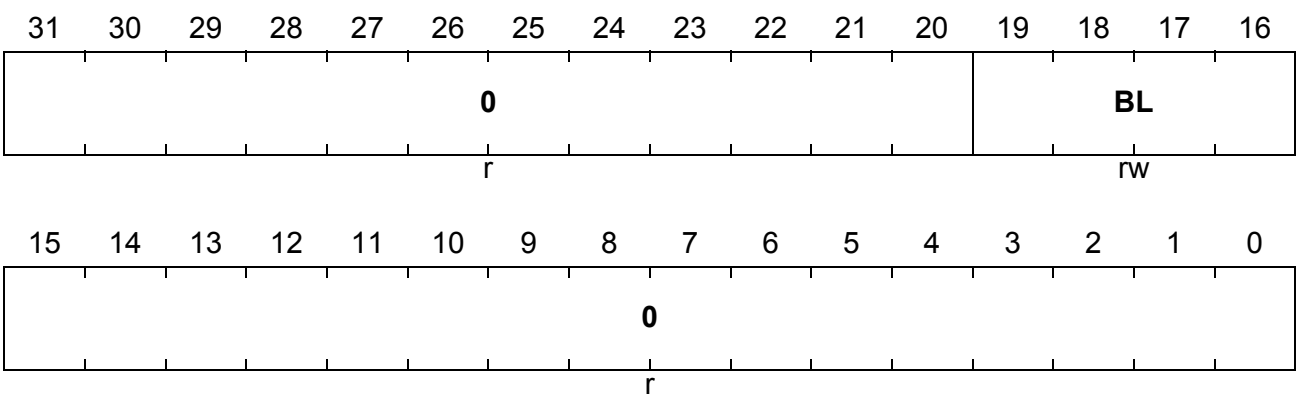
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
LBE	1	rw	**Little/Big Endian Byte Swap** 0 Little Endian 1 Big Endian
0	0, [31:2]	r	**Reserved;** read as 0; should be written with 0.

DTCONF3
DT Configuration 3 Register

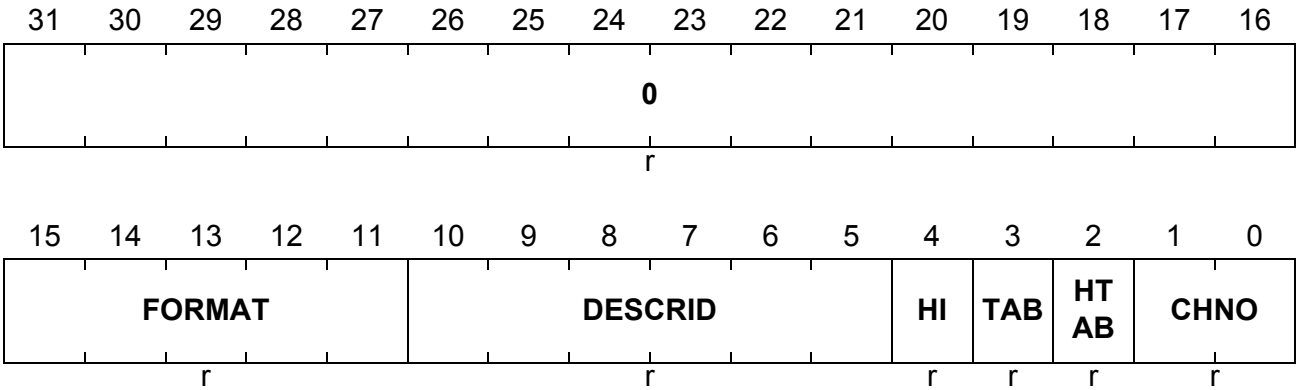
Reset Value: 0009 0000$_H$



Field	Bits	Type	Description
BL	[19:16]	rw	**Burst Length** This value must be set to 0010_B.
0	[15:0], [31:20]	r	**Reserved;** read as 0; should be written with 0.

DTISFIFO
DT Interrupt Status FIFO

Reset Value: 0000 0000$_H$

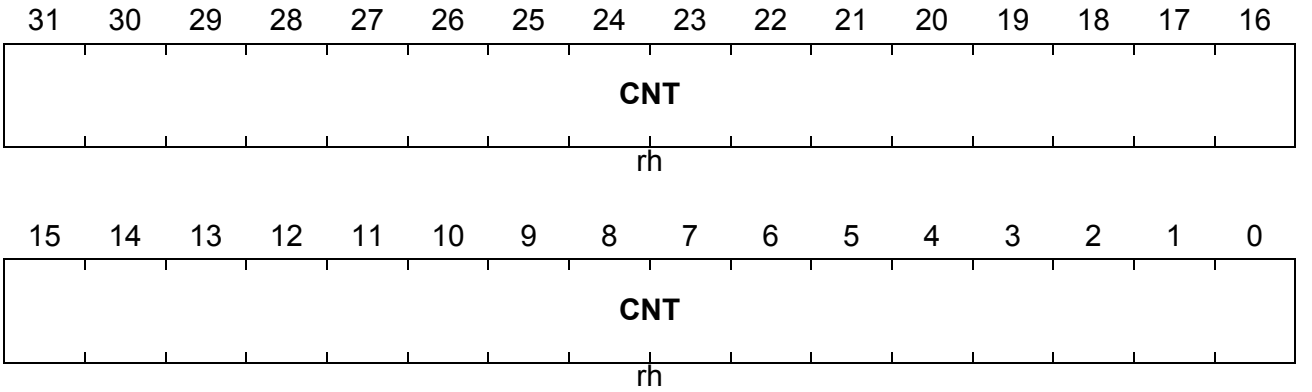


Field	Bits	Type	Description
CHNO	[1:0]	r	**Channel Number** Identifies the channel number of the channel that has generated the interrupt.
HTAB **CMDC**	2	r	**Hold Caused Transmit Abort** (FORMAT = ‘10111’) Issued if the DMUT retrieves a transmit descriptor where HOLD = 1 and FE = 0. The interrupt will be generated after the data section is transferred completely. **Command Completed** (FORMAT = ‘10010’) Identifies a successful channel command performed by the host CPU. A new command can be given by the CPU for this channel. Reserved when the register works as Command Interrupt Vector specified by FORMAT bits.
TAB	3	r	**Transmit Abort** Set when the DMUT detects a transmit abort command and a frame is not transmitted completely for this channel. This bit will also be set when NO = 0 and FE = 0. Reserved when register works as Command Interrupt Vector specified by FORMAT bits.

Field	Bits	Type	Description
HI	4	r	**Host Initiated Interrupt** If the HI bit in the transmit descriptor is set, an interrupt is generated with this bit set when the DMUT finishes the old transmit descriptor and branches to the next transmit descriptor. Reserved when register operates as Command Interrupt Vector specified by FORMAT bits.
DESCRID	[10:5]	r	**Descriptor ID** Identifies the transmit descriptor that generated the interrupt. Reserved except bit 16 when register works as Command Interrupt Vector specified by FORMAT bits.
FORMAT	[15:11]	r	**Format** This pattern specifies the format of the interrupt vector. 10111 Channel Interrupt Vector 10010 Command Interrupt Vector
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

DTFFCR
DT FIFO Full Counter Register

Reset Value: 0000 0000$_H$



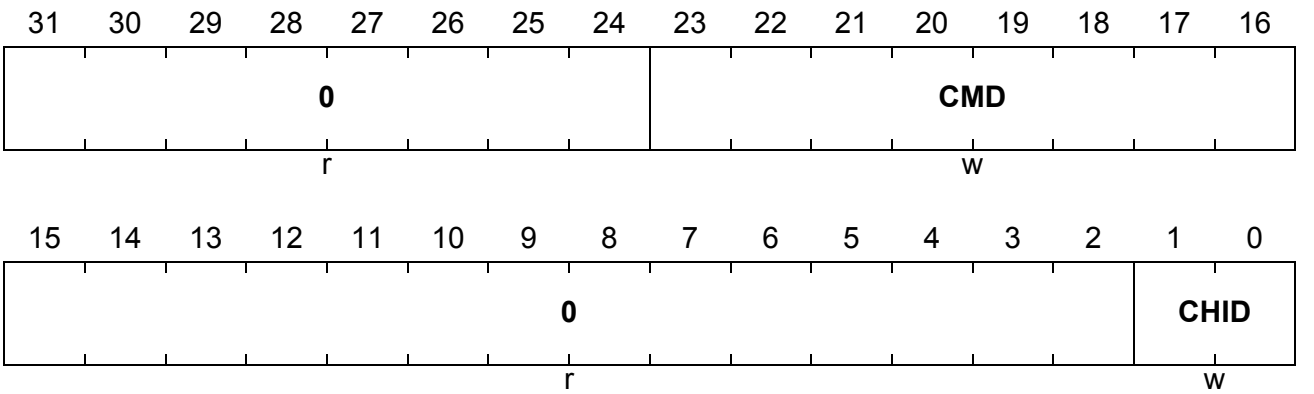
Field	Bits	Type	Description
CNT	[31:0]	rh	**Counter** This counter is incremented when an interrupt occurs and the interrupt status cannot be stored in the interrupt status FIFO due to a FIFO full condition. The DT is halted until at least one interrupt status is read from the interrupt status FIFO. In that case, the pending interrupt status is forwarded to the interrupt status FIFO and DT resumes operation. Thus, this counter indicates how often an interrupt status could not be stored due to a FIFO full condition.

Note: After the software read the value of CNT bit field, the hardware will clear it.

31.2.4 RB Registers

The RB registers are used to configure and control the RB block.

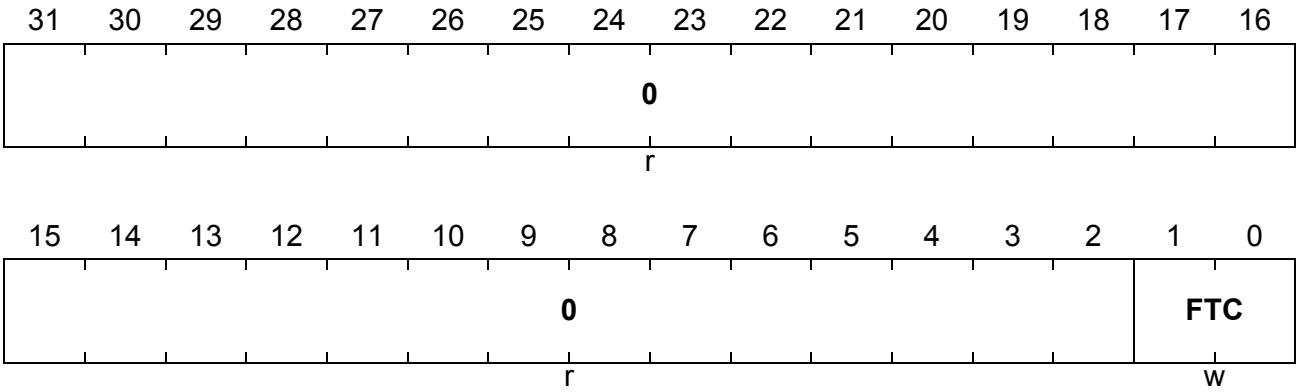
RBCC
RB Channel Command Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CHID	[1:0]	w	**Channel ID** Identifies the channel that is configured. This value must be set to ‘00’.
CMD	[23:16]	w	**Command** 01_H Initialize Others: Reserved
0	[15:2], [31:24]	r	**Reserved;** read as 0; should be written with 0.

RBCBL
RB Channel Burst Length Register

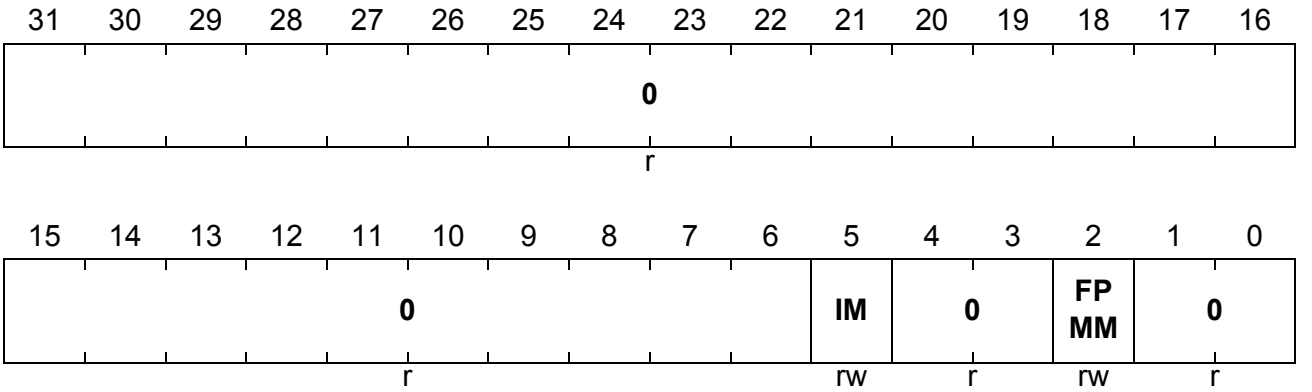
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
FTC	[1:0]	w	**Forward Threshold Code** The receive buffer forward threshold code determines the maximum packet size in WORDS which will be stored in the internal receive buffer for a specific channel before data are forwarded to the DMU. When the packet size reaches the receive buffer forward threshold or a packet has been completely received (FE), the packet will be forwarded to system memory. FTC Threshold in WORDS 10 8 Others: Reserved
0	[31:2]	r	**Reserved;** read as 0; should be written with 0.

RBFPM
RB Free Pool Monitor Register

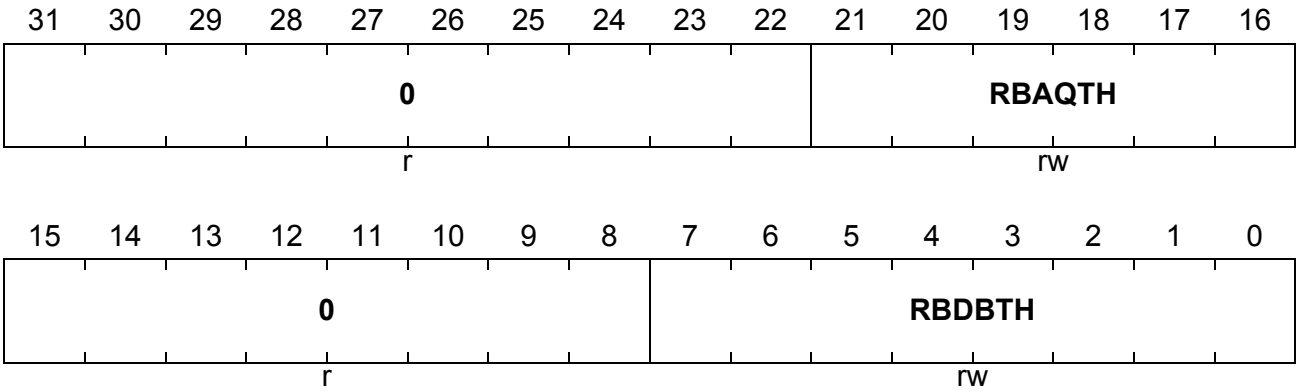
Reset Value: 0000 0020$_H$



Field	Bits	Type	Description
FPMM	2	rw	**Free Pool Monitor Mode** 0 Capture free pool count. It is reflected in Free Pool Count Register 1 Interrupt, if free pool falls below programmed threshold.
IM	5	rw	**Interrupt Mask** 0 Enable interrupt 1 Disable interrupt
0	[1:0], [4:3], [31:6]	r	**Reserved;** read as 0; should be written with 0.

RBFPTH
RB Free Pool Threshold Register

Reset Value: 0020 0001$_H$



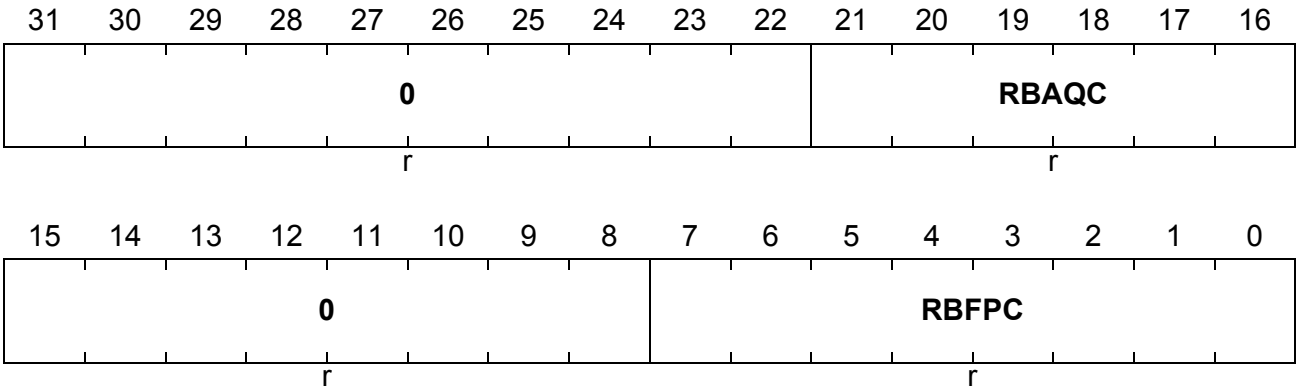
Field	Bits	Type	Description
RBDBTH	[7:0]	rw	**RB Free Pool Threshold** Set value: 03_H to FF_H
RBAQTH	[21:16]	rw	**RB Free Action Queue Threshold** Set value: 01_H to 20_H
0	[15:8], [31:22]	r	**Reserved;** read as 0; should be written with 0.

FF_H is the maximum value of RBDBTH bit, which means free pool threshold is set as the maximum number of locations (256 × 4 Bytes) of receive buffer. With this setting, the interrupt Ethernet_RBSRC0 (if enabled) is activated every time once the RB has data. RBFPC bit is used to monitor the value of free buffer locations inside the receive buffer.

Similarly, 20_H is the maximum value of RBAQTH bit. With this setting, the interrupt Ethernet_RBSRC1 (if enabled) is activated every time once the RB has an action. RBAQC bit is used to monitor the value of free locations of action queue inside the receive buffer.

RBFPCNT
RB Free Pool Count Register

Reset Value: 0020 00FF$_H$

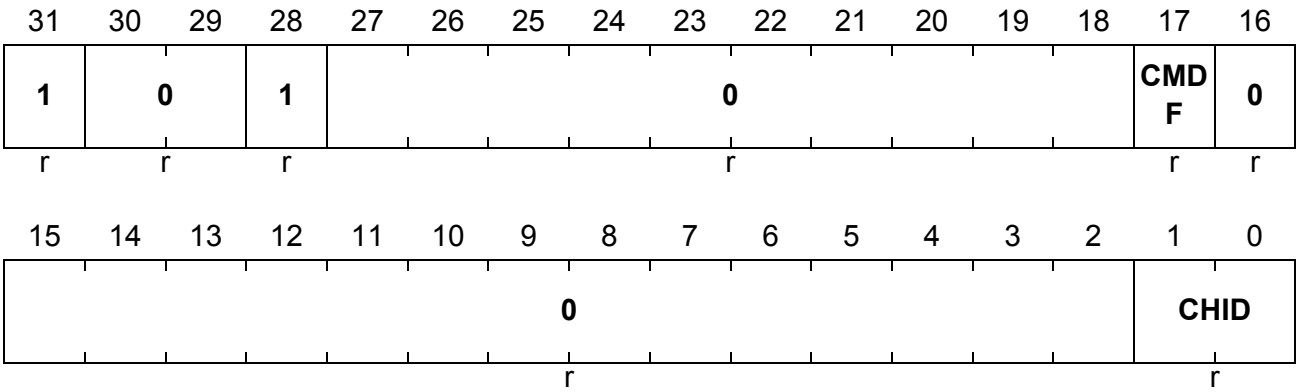


Field	Bits	Type	Description
RBFPC	[7:0]	r	**RB Free Pool Count** This value determines the actual number of free buffer locations inside the receive buffer. After reset, a total number of 256 receive buffer locations, which equals the 1 Kbyte Transmit Buffer, is available.
RBAQC	[21:16]	r	**RB Free Action Queue Count** This value determines that actual number of free locations of action queue inside the receive buffer.
0	[15:8], [31:22]	r	**Reserved;** read as 0; should be written with 0.

31.2.5 TB Registers

The TB registers are used to configure and control the TB block.

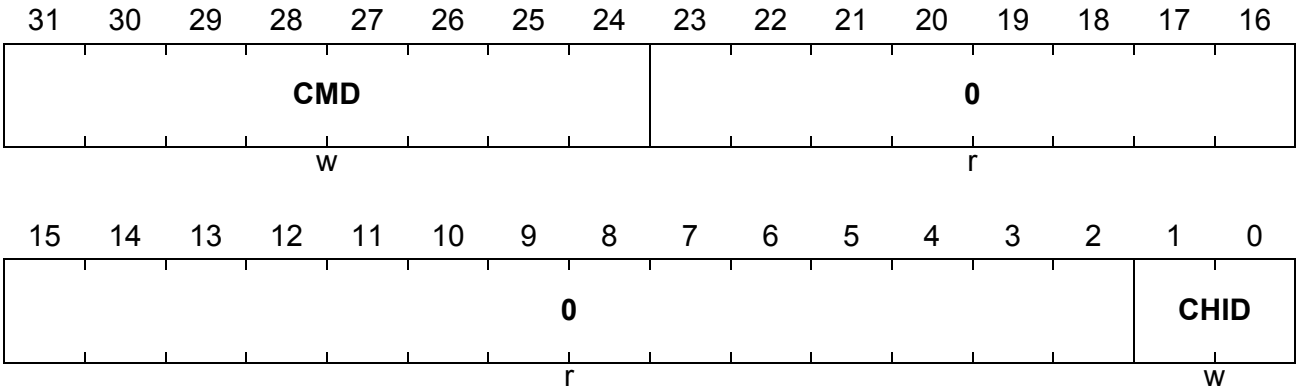
TBISR
TB Interrupt Status Register **Reset Value: 9002 0000$_H$**



Field	Bits	Type	Description
CHID	[1:0]	r	**Channel ID** Identifies the channel that is selected; this value must be set to ‘00’.
CMDF	17	r	**Command Failed** Initialization goes wrong.
0	[16:2] [27:18] [30:29]	r	**Reserved;** read as 0; should be written with 0.
1	31, 28	r	**Reserved**

TBCC
TB Channel Command Register

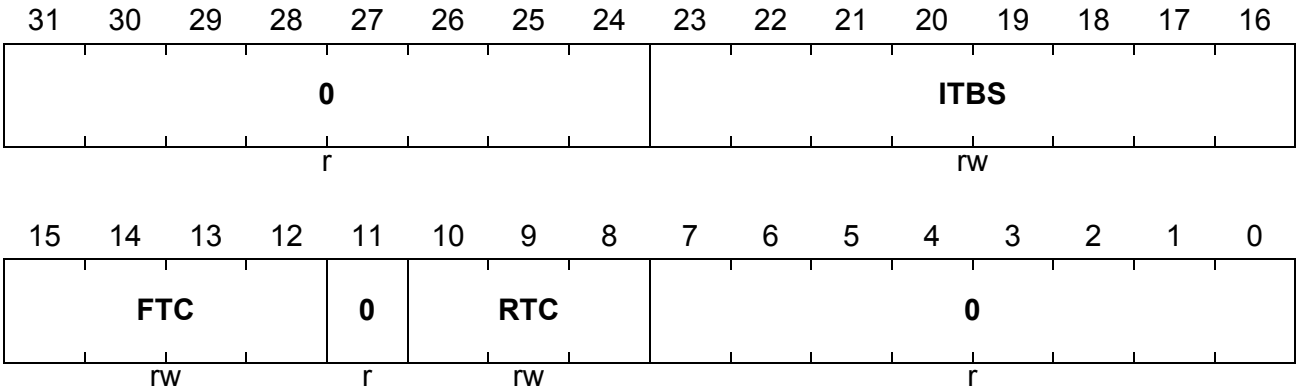
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
CHID	[1:0]	w	**Channel ID** Identifies the channel that is selected; this value must be set to ‘00’.
CMD	[31:24]	w	**Command** 01_H Init 02_H Off Others: Reserved
0	[23:2]	r	**Reserved;** read as 0; should be written with 0.

TBCPR
TB Channel Parameter Register

Reset Value: 0020 0000$_H$



Field	Bits	Type	Description
RTC	[10:8]	rw	**Refill Threshold Code** The internal Transmit Buffer has a programmable number of buffer locations per channel. When the number of free locations reach the Transmit Buffer refill threshold, the internal Transmit Buffer requests new data from the DMU. RTC Threshold in WORDS 000 1 001 4 010 8 011 12 100 16 101 24 110 32 111 40

Field	Bits	Type	Description
FTC	[15:12]	rw	**Forward Threshold Code** The Transmit Buffer forward threshold code determines the number of buffer locations which must be filled before protocol machines (e.g. Ethernet MAC) start transmission/segmentation. Nevertheless, the Transmit Buffer forwards data packets to the protocol machine as soon as an entire packet or the end of a packet is stored in the Transmit Buffer. FTC Threshold in WORDS 0000 1 0001 4 0010 8 0011 12 0100 16 0101 24 0110 32 0111 40 1000 48 1001 64 1010 96 1011 128 1100 192 1101 254 1111 Reserved
ITBS	[23:16]	rw	**Individual Transmit Buffer Size** The Transmit Buffer size configures the number of internal Transmit Buffer locations for a particular channel. Buffer locations will be allocated on command transmit init and released after command transmit off. *Note: The internal architecture is 32 bits wide. Therefore, each buffer location corresponds to four data octets.* *Note: The sum of the transmit forward threshold and the transmit refill threshold must be smaller than the internal buffer size.*
0	[7:0], 11, [31:24]	r	**Reserved;** read as 0; should be written with 0.

For the product that has no high speed memory, FTC bit is suggested to set the highest value “1101”, which can avoid the separation of long frame when “under run” of TX occurs. For short frame, it could reduce the threshold value according to frame length, but not less than the frame length.

RTC may be set to “000”, which means the data is required once the buffer has free space.

31.3 Ethernet Controller Implementation

This section describes Ethernet Controller module interfaces with the clock control, port connections, interrupt control, and address decoding.

31.3.1 Interfaces of the Ethernet Controller Module

Figure 31-17 shows the TC1130 specific implementation details and interconnections of the Ethernet Controller module. The Ethernet Controller module has 18 I/O lines that are connected with Port 1 and three specific pins. The Ethernet Controller is further supplied by a separate interrupt control, address decoding, and port control logic.

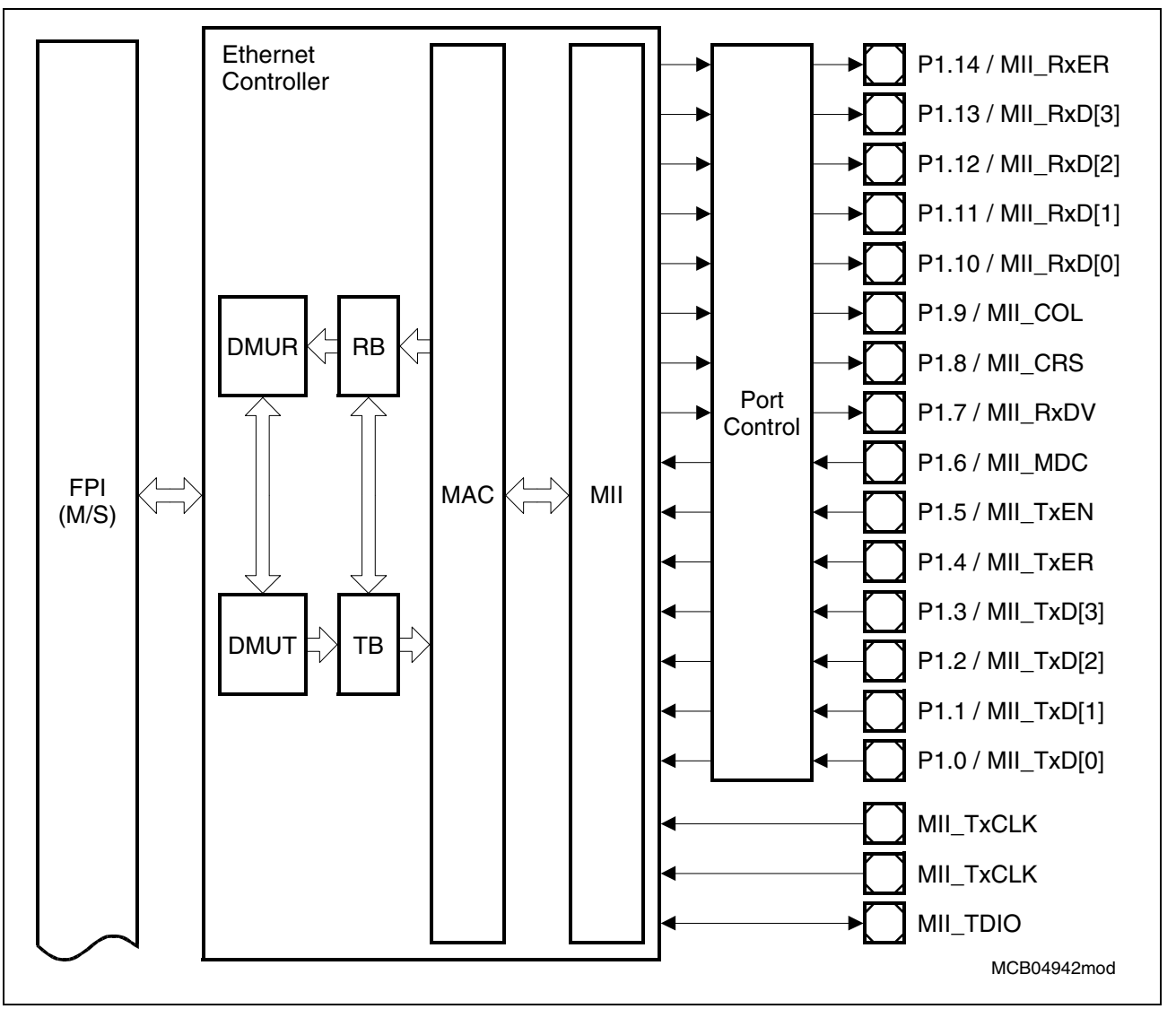


Figure 31-17 Ethernet Controller Module Implementation and Interconnections

31.3.2 External Ethernet Controller Module Registers

Figure 31-18 summarizes the module related external registers required for Ethernet Controller programming (see also **Figure 31-16** for the module kernel-specific registers).

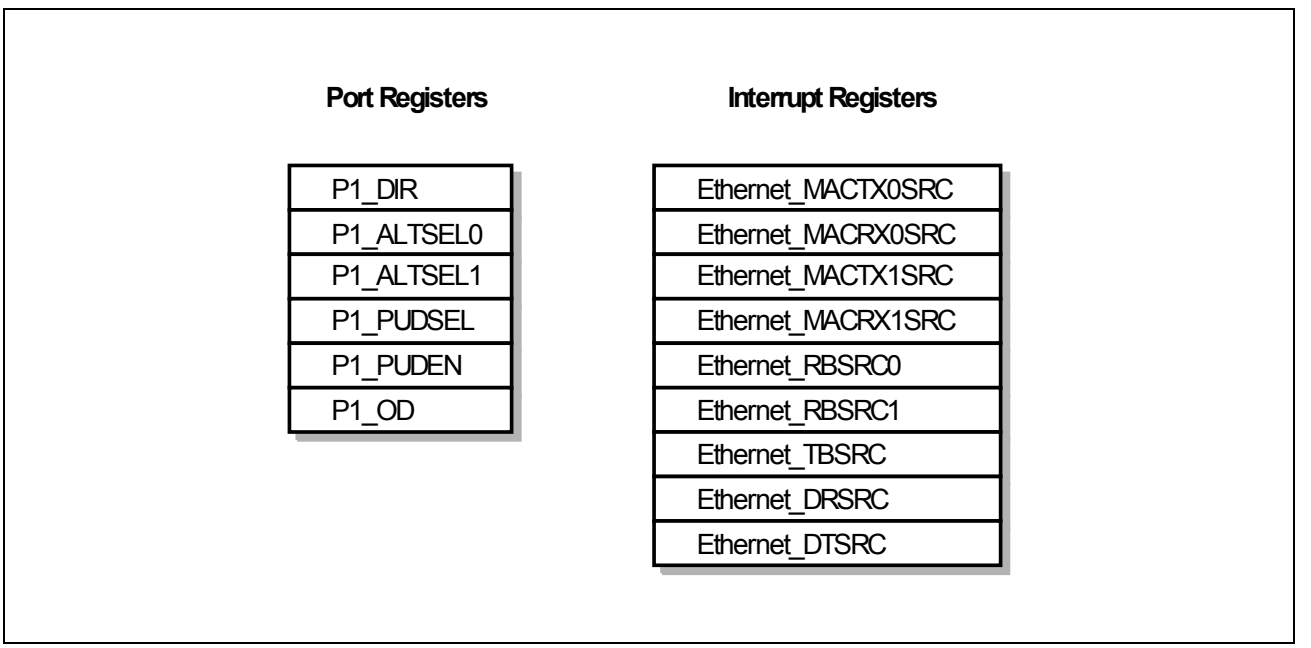


Figure 31-18 Ethernet Controller Implementation Specific Special Function Registers

31.3.2.1 Port Control Registers

The interconnections between the Ethernet Controller and the port I/O lines are controlled in the port logics. The following port control operations selections must be executed:

- Input/output function selection (DIR register)
- Alternate function selection (ALTSEL0 and ALTSEL1 registers)
- Input/Output driver characteristic control (PUDSEL, PUDEN and OD registers)

Input/Output Function Selection

The port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as pull-up/down devices, port direction (input/output), open-drain, and alternate output selections. The I/O lines for the Ethernet Controller are controlled by the port input/output control registers of Port1.

Table 31-11 shows how bits and bit fields must be programmed for the required I/O functionality of the Ethernet Controller I/O lines. This table also shows the values of the peripheral input select registers.

Table 31-11 Ethernet Controller I/O Line Selection and Setup

Module	Port Lines	Input/Output Control Register Bits	I/O
Ethernet Controller	P1.0/MII_TXD0	P1_DIR.P0 = 1_B	Output
		P1_ALTSEL0.P0 = 1_B	
		P1_ALTSEL1.P0 = 0_B	
	P1.1/MII_TXD1	P1_DIR.P1 = 1_B	Output
		P1_ALTSEL0.P1 = 1_B	
		P1_ALTSEL1.P1 = 0_B	
	P1.2/MII_TXD2	P1_DIR.P2 = 1_B	Output
		P1_ALTSEL0.P2 = 1_B	
		P1_ALTSEL1.P2 = 0_B	
	P1.3/MII_TXD3	P1_DIR.P3 = 1_B	Output
		P1_ALTSEL0.P3 = 1_B	
		P1_ALTSEL1.P3 = 0_B	
	P1.4/MII_TXER	P1_DIR.P4 = 1_B	Output
		P1_ALTSEL0.P4 = 1_B	
		P1_ALTSEL1.P4 = 0_B	
	P1.5/MII_TXEN	P1_DIR.P5 = 1_B	Output
		P1_ALTSEL0.P5 = 1_B	
		P1_ALTSEL1.P5 = 0_B	
	P1.6/MII_MDC	P1_DIR.P6 = 1_B	Output
		P1_ALTSEL0.P6 = 1_B	
		P1_ALTSEL1.P6 = 0_B	
	P1.7/MII_RXDV	P1_DIR.P7 = 0_B	Input
	P1.8/MII_CRS	P1_DIR.P8 = 0_B	Input
	P1.9/MII_COL	P1_DIR.P9 = 0_B	Input
	P1.10/MII_RXD0	P1_DIR.P10 = 0_B	Input
	P1.11/MII_RXD1	P1_DIR.P11 = 0_B	Input
	P1.12/MII_RXD2	P1_DIR.P12 = 0_B	Input
	P1.13/MII_RXD3	P1_DIR.P13 = 0_B	Input
	P1.14/MII_RXER	P1_DIR.P14 = 0_B	Input

P1_DIR
Port 1 Direction Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-14)**	n	rw	**Port 1 Pin 0 - 14 Direction Control**[1)] 0 Direction is set to input (default after reset) 1 Direction is set to output
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for Ethernet Controller I/O port control.

P1_ALTSELn (n = 0, 1)
Port 1 Alternate Select Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 31-12 Function of the Bits P1_ALTSEL0.Pn and P1_ALTSEL1.Pn (n = 0-6)[1)]

P1_ALTSEL0.Pn	P1_ALTSEL1.Pn	Function
1	0	Alternate Select 1

1) Shaded bits and bit field are don't care for Ethernet Controller I/O port control.

The Ethernet Controller ports also offer the possibility to configure the following output characteristics:

- Push/pull (optional pull-up/pull-down)
- Open drain with internal pull-up
- Open drain with external pull-up

P1_PUDSEL
Port 1 Pull-Up/Pull-Down Select Register

Reset Value: 0000 FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-14)**	n	rw	**Pull-Up/Pull-Down Select Port 1 Bit n**[1)] 0 Pull-down device is selected 1 Pull-up device is selected
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for Ethernet Controller I/O port control.

P1_PUDEN
Port 1 Pull-Up/Pull-Down Enable Register

Reset Value: 0000 FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-14)**	n	rw	**Pull-Up/Pull-Down Enable at Port 1 Bit n**[1)] 0 Pull-up or Pull-down device is disabled 1 Pull-up or Pull-down device is enabled
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for Ethernet Controller I/O port control.

P1_OD
Port 1 Open Drain Control Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	**P14**	**P13**	**P12**	**P11**	**P10**	**P9**	**P8**	**P7**	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Pn **(n = 0-14)**	n	rw	**Port 1 Pin n Open Drain Mode**[1)] 0 Normal Mode, output is actively driven for 0 and 1 state 1 Open Drain Mode, output is actively driven only for 0 state
0	[31:16]	r	**Reserved;** read as 0; should be written with 0.

1) Shaded bits and bit field are don't care for Ethernet Controller I/O port control.

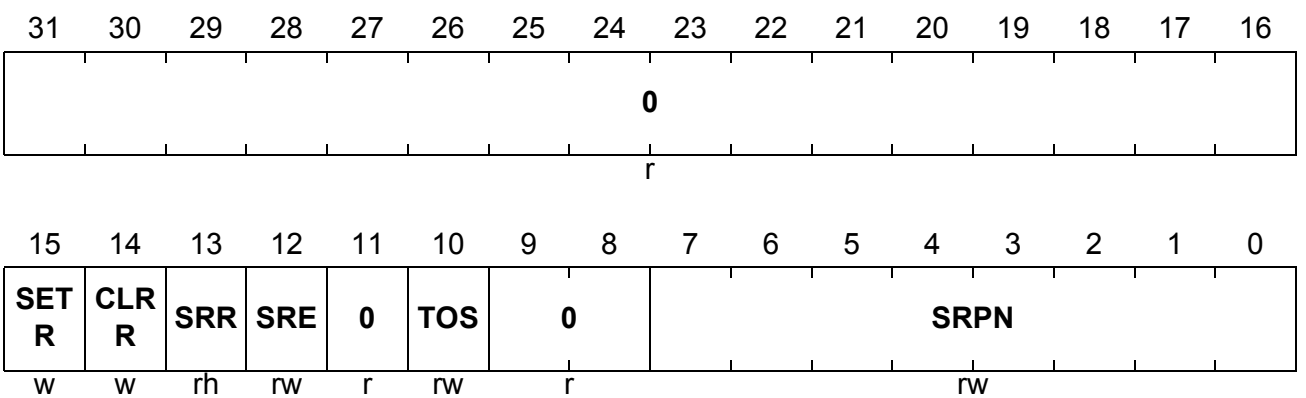
31.3.3 Interrupt Registers

The nine interrupts of the Ethernet Controller are controlled by the following service request control registers:

- Ethernet_DTSRC: controls the DMUT interrupt
- Ethernet_DRSRC: controls the DMUR interrupt
- Ethernet_TBSRC: controls the TB interrupt
- Ethernet_RBSRC0: controls the Data Buffer Free Pool interrupt of RB
- Ethernet_RBSRC1: controls the Action Queue Free Pool interrupt of RB
- Ethernet_MACTX0SRC: controls the MAC Transmit interrupt
- Ethernet_MACRX0SRC: controls the MAC Receive interrupt
- Ethernet_MACTX1SRC: controls the MAC Transmit interrupt
- Ethernet_MACRX1SRC: controls the MAC Receive interrupt

Ethernet_DTSRC
Ethernet DMUT Interrupt Service Request Control Register
Ethernet_DRSRC
Ethernet DMUR Interrupt Service Request Control Register
Ethernet_TBSRC
Ethernet TB Interrupt Service Request Control Register
Ethernet_RBSRC0
Ethernet RB Interrupt Service Request Control Register 0
Ethernet_RBSRC1
Ethernet RB Interrupt Service Request Control Register 1
Ethernet_MACTX0SRC
Ethernet MAC Transmit Interrupt Service Request Control Register 0
Ethernet_MACRX0SRC
Ethernet MAC Receive Interrupt Service Request Control Register 0
Ethernet_MACTX1SRC
Ethernet MAC Transmit Interrupt Service Request Control Register 1
Ethernet_MACRX1SRC
Ethernet MAC Receive Interrupt Service Request Control Register 1

Reset Values: 0000 0000$_H$



Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number**
TOS	10	rw	**Type of Service Control**
SRE	12	rw	**Service Request Enable**
SRR	13	rh	**Service Request Flag**
CLRR	14	w	**Request Clear Bit**
SETR	15	w	**Request Set Bit**
0	[9:8], 11, [31:16]	r	**Reserved;** read as 0; should be written with 0.

Note: Additional details on interrupt handling and processing are provided in the "Interrupt System" chapter of the TC1130 System Units User's Manual.

31.3.4 Ethernet Controller Register Address Range

In the TC1130, the registers of the Ethernet Controller Module are located in the following address range:

- Module Base Address: F200 0100_H
  Module End Address: F200 $05FF_H$
- Absolute Register Address = Module Base Address + Offset Address
  (offset addresses see **Table 31-10**)

Note: The complete and detailed address map of the Ethernet Controller module is described in the chapter "Register Overview" of the TC1130 System Units User's Manual.

Keyword Index

This User's Manual consists of two volumes, "System Units" [1] and "Peripheral Units" [2]. For your convenience, this keyword index (and also the table of contents and the register index) lists both volumes, so you can immediately find the reference to the desired section in the corresponding document ([1] or [2]).

A

Absolute
  Difference 2-42 [1]
  Value 2-42 [1]
Acronyms 1-4 [1]
Addition instructions 2-40 [1]
Address
  Arithmetic 2-56 [1]
  Comparison 2-57 [1]
  Return 2-14 [1]
Address maps
  Segment 15 7-3 [1]
Address register 2-14 [1], 2-56 [1], 2-60 [1]
ALN
  Alignment 16-10 [1]
Architecture
  Traps 16-2 [1]
Arithmetic
  Instructions 2-39 [1], 2-48 [1]
  Operations 2-56 [1]
ASC
  Address ranges 23-60 [2]
  Asynchronous mode
  23-5 [2]–23-17 [2]
    Data frames 23-6 [2]–23-7 [2]
    Data path selection 23-17 [2]
  Baud rate generation
  23-20 [2]–23-25 [2]
    Asynchronous modes 23-21 [2]
    Synchronous mode 23-24 [2]
  Block diagram
    Asynchronous modes 23-5 [2]
    Synchronous mode 23-18 [2]
  DMA request outputs 23-60 [2]
  Error detection 23-25 [2]
  Features 23-3 [2]
  Interrupt generation 23-27 [2]
  IrDA Mode
    Function 23-15 [2]
    IrDA frames 23-8 [2]
  Module implementation
  23-43 [2]–23-60 [2]
    DMA request outputs 23-60 [2]
    Input/output function selection
    23-49 [2]
    Interrupt registers 23-57 [2]
    Module clock control 23-45 [2]
    Peripheral input select 23-47 [2]
  Registers 23-29 [2]–23-41 [2]
    Address ranges 23-60 [2]
    BG **23-34 [2]**
    CON **23-30 [2]**
    FDV **23-35 [2]**
    FSTAT **23-41 [2]**
    Offset addresses 23-29 [2]
    Overview 23-29 [2]
    PISEL **23-30 [2]**
    PMW **23-36 [2]**
    RBUF **23-37 [2]**
    RXFCON **23-38 [2]**
    TBUF **23-36 [2]**
    TXFCON **23-40 [2]**
    WHBCON **23-33 [2]**
  Synchronous mode
  23-18 [2]–23-19 [2]
    Timings 23-20 [2]
ASI
  Address Space Identifier 10-5 [1]
Asynchronous trap 16-5 [1]

B

Base
Pointers 2-56 [1]
Basic data types 2-62 [1]
BCU 18-21 [1]
Bit
Enable / Disable 2-27 [1]
Field 2-63 [1]
Operations 2-54 [1]
Bit-field extract instructions 2-44 [1]
Block diagram 1-10 [1]
Boolean equations 2-55 [1]
Boot
Debug boot 5-13 [1]
Normal boot 5-12 [1]
Selection table 5-12 [1]
Boot operation 5-11 [1]–5-14 [1]
Branch instructions 2-58 [1]
Burst mode timings 14-80 [1]
Bus System 18-1 [1]–18-43 [1]

C

C
Cacheability bit 10-5 [1]
CAN
Block diagram 30-4 [2]
Clock control 30-84 [2]
DMA requests 30-101 [2]
Interrupt structure 30-86 [2]
Module implementation 30-87 [2]–30-101 [2]
Input/output function selection 30-93 [2]
MultiCAN
Analysis mode 30-15 [2]
Bit timing 30-7 [2]
Block diagram 30-5 [2]
Error handling 30-9 [2]
Gateway mode 30-37 [2]
Interrupts 30-10 [2]
Message acceptance filtering 30-18 [2]
Message object FIFO 30-32 [2]
Message object lists 30-11 [2]
Node control 30-7 [2]
Registers
LISTi **30-48 [2]**
MCR **30-45 [2]**
MITR **30-46 [2]**
MOAMRn **30-79 [2]**
MOARn **30-80 [2]**
MOCTRn **30-68 [2]**
MODATAHn **30-84 [2]**
MODATALn **30-83 [2]**
MOFCRn **30-74 [2]**
MOFGPRn **30-78 [2]**
MOIPRn **30-73 [2]**
MSIDk **30-49 [2]**
MSIMASK **30-50 [2]**
MSPNDk **30-49 [2]**
NBTRx **30-61 [2]**
NCRx **30-51 [2]**
NECNTx **30-63 [2]**
NFCRx **30-64 [2]**
NIPRx **30-59 [2]**
NPCRx **30-60 [2]**
NSRx **30-55 [2]**
Offset addresses 30-41 [2]
Overview 30-40 [2]
PANCTR **30-42 [2]**
CBS_COMDATA **21-54 [1]**
CCPN
Current CPU Priority Number 2-27 [1]
CCU6 1-25 [1]
DMA request outputs 29-111 [2]
Module implementation
DMA request outputs 29-111 [2]
Module clock control 29-95 [2]
Registers
Address ranges 29-111 [2]
CC63R **29-60 [2]**
CC63SR **29-61 [2]**
CC6xR **29-54 [2]**
CC6xSR **29-55 [2]**
CMPMODIF **29-43 [2]**

CMPSTAT **29-41 [2]**
IEN **29-85 [2]**
INP **29-89 [2]**
IS **29-78 [2]**
ISR **29-83 [2]**
ISS **29-81 [2]**
MCMCTR **29-72 [2]**
MCMOUT **29-69 [2]**
MCMOUTS **29-68 [2]**
MODCTR **29-62 [2]**
Offset addresses 29-36 [2]
PISEL **29-38 [2]**, **29-40 [2]**
PSLR **29-66 [2]**
T12 **29-52 [2]**
T12DTC **29-56 [2]**
T12MSEL **29-74 [2]**
T12PR **29-53 [2]**
T13 **29-58 [2]**
T13PR **29-59 [2]**
TCTR0 **29-44 [2]**
TCTR2 **29-47 [2]**
TCTR4 **29-50 [2]**
TRPCTR **29-64 [2]**
CDO 16-12 [1]
CDU 16-12 [1]
CLC register 3-23 [1]
Clock operation
Registers
Address range 3-5 [1]
Offset addresses 3-5 [1]
Clock system 3-1 [1], 3-1 [1]–3-35 [1]
CGU block diagram 3-3 [1]
Clock source 3-17 [1]
Clock tree diagram 3-2 [1]
Main oscillator 3-6 [1]
Module clock generation 3-22 [1]
CLC register 3-23 [1]
Fractional divider 3-28 [1]
Module implementation 3-35 [1]
OSC_CON **3-8 [1]**
Oscillator run detection 3-7 [1]
Compare instruction 2-51 [1], 2-53 [1]
Comparison instructions 2-56 [1], 2-57 [1]
comparison instructions 2-57 [1]
Conditional
Branch instructions 2-59 [1]
Expressions 2-52 [1]
Instructions 2-42 [1]
Jump instructions 2-59 [1]
Configuration inputs
Register 5-15 [1]
Context 2-7 [1]
Instructions 2-64 [1]
Loading 2-65 [1]
Lower 2-15 [1]
Restoring 2-64 [1]
Saving 2-64 [1]
Storing 2-65 [1]
Upper 2-15 [1]
Core
Accesses 2-66 [1]
Special Function Register 2-12 [1]
CPS
Registers
CPU_SBSRC **2-80 [1]**
CPU_SRCx **2-82 [1]**
CPU 2-1 [1]
Address ranges 2-83 [1]
Core SFRs
BIV **2-29 [1]**
BTV **2-30 [1]**
FCX **2-23 [1]**
ICR **2-27 [1]**
ISP **2-26 [1]**
LCX **2-25 [1]**
PC **2-17 [1]**
PCX **2-24 [1]**
PCXI **2-21 [1]**
PSW **2-17 [1]**, **2-18 [1]**
Registers
Address ranges 2-83 [1]
Service request nodes 15-22 [1]
CPU architecture overview 2-1 [1]
Processor registers 2-12 [1]
Context management registers 2-23 [1]

Debug registers 2-32 [1]
Interrupt/trap control registers 2-27 [1]
Memory protection registers 2-32 [1]
Program state registers 2-15 [1]
Stack registers 2-26 [1]
System control register 2-31 [1]
CPU slave interface, see “CPS” 2-80 [1]
CPU SRAMs
Bit chain control register 11-11 [1]
Bit chain data in register 11-12 [1]
Bit chain data out register 11-13 [1]
Bit chain structure 11-5 [1]
Configuration programming 11-4 [1]
Soft-error trapped address register 11-13 [1]
SRAM redundancy address register n 11-10 [1]
CSA
Context Save Area 2-8 [1]
CSFRs 2-66 [1]
CSU 16-12 [1]
CTYP 16-12 [1]

D

DAE 16-10 [1], 16-13 [1]
Data
General Purpose Registers 2-14 [1]
Memory 2-66 [1]
Register 2-14 [1], 2-57 [1]
Data memory interface 9-1 [1]–9-7 [1]
Block diagram 9-2 [1]
Registers 9-3 [1]
Address range 9-7 [1]
Offset addresses 9-3 [1]
Overview 9-3 [1]
Data memory unit 11-1 [1]–11-14 [1]
Direct Translation 2-10 [1]
DMA 17-2 [1]
Address update 17-23 [1]
Block diagram 17-6 [1]
Channel request 17-10 [1]
Channel reset 17-22 [1]
Circular buffer 17-17 [1]
Definition of terms 17-3 [1]
Direct Memory Access 2-9 [1]
Features 17-2 [1]
Interrupt generation 17-18 [1], 17-19 [1]
OCDS 17-30 [1]
Operation 17-7 [1]
Operation modes 17-10 [1]
Principle 17-5 [1]
Registers
ADRCR0n **17-66 [1]**
CHCR0n **17-59 [1]**
CHICRmn **17-64 [1]**
CHRSTR **17-41 [1]**
CHSR0n **17-63 [1]**
CLRE **17-49 [1]**
DADR0n **17-71 [1]**
DMA_TOCTR **17-88 [1]**
EER **17-45 [1]**
ERRSR **17-47 [1]**
GINTR **17-40 [1]**
HTREQ **17-44 [1]**
INTCR **17-53 [1]**
INTSR **17-51 [1]**
ME0AENR **17-56 [1]**
ME0ARR **17-58 [1]**
ME0PR **17-56 [1]**
ME0R **17-55 [1]**
MESR **17-54 [1]**
OCDSR **17-36 [1]**
Overview **17-34 [1]**
SADR0n **17-70 [1]**
SHADR0n **17-72 [1]**
STREQ **17-43 [1]**
SUSPMR **17-38 [1]**
TRSR **17-42 [1]**
WRPSR **17-52 [1]**
Request assignment 17-29 [1]
Shadow registers 17-7 [1]
Switch to MLIs and FPI buses 17-26 [1]
Transaction control 17-25 [1]

DMA, see “Data Memory Access Controller”
DMI
Registers
DMI_ATR **9-6 [1]**
DMI_CON **9-4 [1]**
DMI_STR **9-5 [1]**
DMI, see “Data memory interface”
DMU
Address ranges 11-14 [1]
Registers 11-7 [1]
Address ranges 11-14 [1]
CSCACTL **11-11 [1]**
CSCADIN **11-12 [1]**
CSCADOUT **11-13 [1]**
Offset addresses 11-8 [1]
Overview 11-7 [1]
SETA **11-13 [1]**
SRARn **11-10 [1]**
DMU, see “Data memory unit”
Document
Acronyms 1-4 [1]
Structure 1-1 [1]
Terminology 1-3 [1]
Textual conventions 1-1 [1]
DSE 16-13 [1]

E

EBU
Module implementation
14-144 [1]–14-154 [1]
Input/output function selection 14-145 [1]
Pull-up control 4-8 [1]
EBU, see “External bus interface unit”
Endinit function 20-3 [1]
Ethernet
Block diagram 31-109 [2]
Buffer management 31-21 [2]
Data Management Unit 31-8 [2]
Interrupt 31-46 [2]
Introduction 31-2 [2]
MAC controller 31-25 [2]
MII 31-4 [2]
Module implementation
31-109 [2]–31-117 [2]
Input/output function selection 31-110 [2]
Operation 31-35 [2]
Registers
Address range **31-117 [2]**
DRCMD **31-86 [2]**
DRCONF **31-90 [2]**
DRFFCR **31-85 [2]**
DRFRDA **31-88 [2]**
DRIMR **31-89 [2]**
DRISFIFO **31-82 [2]**
DRMOD **31-88 [2]**
DTCMD **31-91 [2]**
DTCONF **31-95 [2]**
DTCONF3 **31-95 [2]**
DTFFCR **31-98 [2]**
DTFTDA **31-93 [2]**
DTIMR **31-94 [2]**
DTISFIFO **31-96 [2]**
MACCAMADDR **31-61 [2]**
MACCAMCTRL0 **31-53 [2]**
MACCAMCTRL1 **31-62 [2]**
MACCAMDATA **31-61 [2]**
MACCTRL **31-52 [2]**
MACMERRCNT **31-63 [2]**
MACPSECNT **31-64 [2]**
MACRPSECNT **31-65 [2]**
MACRX0IMR **31-76 [2]**
MACRX0ISR **31-74 [2]**
MACRX1IMR **31-80 [2]**
MACRX1ISR **31-78 [2]**
MACRXCTRL **31-57 [2]**
MACRXSTAT **31-58 [2]**
MACSMCTRL **31-60 [2]**
MACSMDATA **31-59 [2]**
MACTX0IMR **31-66 [2]**
MACTX0ISR **31-68 [2]**
MACTX1IMR **31-70 [2]**
MACTX1ISR **31-72 [2]**
MACTXCTRL **31-54 [2]**

MACTXSTAT **31-55 [2]**
Offset addresses 31-50 [2]
Overview 31-49 [2]
RBCBL **31-100 [2]**
RBCC **31-99 [2]**
RBFPCNT **31-103 [2]**
RBFPM **31-101 [2]**
RBFPTH **31-102 [2]**
TBCC **31-105 [2]**
TBCPR **31-106 [2]**
TBISR **31-104 [2]**
Ethernet controller 1-30 [1]
Exception instruction 2-67 [1]
Extended-size registers 2-14 [1]
External bus interface unit
14-1 [1]–14-154 [1]
Address region parameters 14-37 [1]
Address region selection 14-33 [1], 14-34 [1]
Basic access timing 14-44 [1]
Demultiplexed mode 14-54 [1]
Multiplexed mode 14-59 [1]
Standard access phases 14-44 [1]
Basic operation 14-4 [1]
Block diagram 14-1 [1]
Data width 14-43 [1]
EBU address region 14-32 [1]
EBU register address range 14-154 [1]
Emulation support 14-29 [1]
Emulation boot 14-29 [1]
Overlay memory 14-29 [1]
Example configuration 14-5 [1]
External bus arbitration 14-12 [1]
Modes 14-14 [1]
Signals 14-12 [1], 14-13 [1]
Features 14-3 [1]
Instruction fetches 14-67 [1]
Internal to external operation 14-32 [1]
Overview 14-2 [1]
Registers 14-109 [1]
Address range 14-154 [1]
ADDRSELx **14-113 [1]**
BFCON **14-132 [1]**
BUSAPx **14-115 [1]**
BUSCONx **14-115 [1]**
CLC **14-112 [1]**
CON **14-130 [1]**
EMUAS **14-121 [1]**
EMUBAP **14-121 [1]**
EMUBC **14-121 [1]**
EMUOVL **14-121 [1]**
Offset addresses 14-109 [1]
Overview 14-109 [1]
SDRMCONx **14-136 [1]**
SDRMODx **14-136 [1]**
SDRMREFx **14-136 [1]**
SDRSTATx **14-136 [1]**
SDRAM interface 14-88 [1]
Bank precharge 14-101 [1]
External interface 14-90 [1]
Initialization sequence 14-92 [1]
Multibanking operation 14-98 [1]
Power down 14-103 [1]
Power up sequence 14-92 [1]
Refresh cycles 14-102 [1]
SDRAM addressing scheme 14-104 [1]
SDRAM burst access timing 14-95 [1]
SDRAM commands 14-90 [1]
Signals 14-89 [1]
Signal description 14-7 [1]
External overlay memory 14-29 [1]
External request unit 15-28 [1]
Block diagram 15-29 [1]
Event trigger logic 15-31 [1]
Features 15-28 [1]
Implementation 15-35 [1]
Interrupt gating logic 15-33 [1]
Registers 15-37 [1]
EICR0 **15-38 [1]**
EICR1 **15-41 [1]**
EIFR **15-44 [1]**
FMR **4-6 [1]**, **4-7 [1]**, **15-45 [1]**
Overview 15-37 [1]
Request select logic 15-30 [1]

F

FCD 16-11 [1]
FCU 16-12 [1]
FDR register 3-31 [1]
Features 1-7 [1]
  CPU 1-8 [1]
  External bus interface 1-8 [1]
  I/O lines 1-9 [1]
  Instruction set 1-8 [1]
  Interrupt system 1-9 [1]
  On-chip memory 1-8 [1]
FFT 2-49 [1]
Foating-point registers 2-14 [1]
FPI
  Bus agent priorities 18-26 [1]
FPI Bus
  Error handling 18-23 [1]
  Starvation protection 18-22 [1]
FPU 2-69 [1]
  Data format 2-69 [1]
  Denormal numbers 2-69 [1]
  Exception handling 2-76 [1]
  Exceptions 2-70 [1]
  Extended precision 2-70 [1]
  Instructions 2-72 [1]
  Interrupt 15-26 [1]
  Rounding 2-71 [1]
Fractional divider 3-28 [1]
  FDR register 3-31 [1]
  Function table 3-33 [1]
  Operating modes 3-28 [1]
  Suspend mode 3-30 [1]

G

General 2-15 [1]
GPIO ports, see “Ports”
GPR 2-12 [1]
GPTU
  Block diagram 28-2 [2]
  Features 28-3 [2]
  Interrupt generation 28-20 [2]
  Module implementation
  28-52 [2]–28-62 [2]
    Input/output function selection
    28-55 [2]
  Output control 28-18 [2]
  Registers 28-22 [2]–28-50 [2]
    Address range 28-62 [2]
    Offset addresses 28-22 [2]
    OSEL **28-47 [2]**
    OUT **28-48 [2]**
    Overview 28-22 [2]
    SRSEL **28-49 [2]**
    T012RUN **28-41 [2]**
    T01IRS **28-24 [2]**
    T01OTS **28-27 [2]**
    T0CBA **28-29 [2]**
    T0DCBA **28-29 [2]**
    T0RCBA **28-30 [2]**
    T0RDCBA **28-30 [2]**
    T1CBA **28-31 [2]**
    T1DCBA **28-31 [2]**
    T1RCBA **28-32 [2]**
    T1RDCBA **28-31 [2]**
    T2 **28-45 [2]**
    T2AIS **28-33 [2]**
    T2BIS **28-35 [2]**
    T2CON **28-38 [2]**
    T2ES **28-36 [2]**
    T2RC0 **28-46 [2]**
    T2RC1 **28-46 [2]**
    T2RCCON **28-43 [2]**
GRWP 16-9 [1]
Guard Bits 2-49 [1]

I

ICACHE 8-4 [1]
ICR
  Interrupt Control Register 2-27 [1]
IIC
  Address range 25-33 [2]
  DMA request outputs 25-33 [2]
  Module implementation
  25-26 [2]–25-33 [2]
    DMA request outputs 25-33 [2]

Input/output function selection 25-29 [2]
Module clock control 25-28 [2]
Registers
Address range 25-33 [2]
BUSCON **25-23 [2]**
CLC **25-28 [2]**
Offset addresses 25-12 [2]
Overview 25-12 [2]
PISEL **25-13 [2]**
RTB **25-25 [2]**
SYSCON **25-14 [2]**
WHBSYSCON **25-20 [2]**
Indexes
Table 2-14 [1]
Instruction
Set architecture 2-39 [1]
Integer arithmetic 2-40 [1]
Interrupt
System disabling 2-67 [1]
System enabling 2-67 [1]
Interrupt system 15-1 [1]–15-54 [1]
Arbitration cycles 15-11 [1]
Arbitration process 15-11 [1]
Block diagram 15-2 [1]
Control register ICR 15-8 [1]
Hints for applications 15-18 [1]–15-22 [1]
Interrupt control unit 15-8 [1]
Interrupt vector table 15-15 [1]
Overview 15-1 [1]
Priorities 15-19 [1]
Service request control register 15-3 [1]
Service request node table 15-52 [1]
Service request nodes 15-3 [1]
Service routine entering 15-12 [1]
Service routine exiting 15-13 [1]
Software initiated 15-22 [1]
Interrupts 2-14 [1]
Special system interrupts
FPU interrupt 15-26 [1]
IOPC 16-10 [1]
ISR
Interrupt Service Routine 2-7 [1]

J

JTAG interface (Cerberus) 21-48 [1]
Block diagram 21-49 [1]
Client instructions 21-56 [1]
Features 21-48 [1]
Registers
CLIENT_ID **21-54 [1]**
COMDATA 21-54 [1]
IOADDR 21-54 [1]
IOCONF **21-50 [1]**
IODATA 21-54 [1]
IOINFO **21-53 [1]**
IOSR **21-52 [1]**
Overview 21-50 [1]
Serial IO lines 21-55 [1]
Shift register 21-58 [1]
Jump instruction 2-58 [1]

L

LBCU
Registers 18-6 [1]
Offset addresses 18-6 [1]
Overview 18-6 [1]
LCX 16-11 [1]
LEA 2-56 [1]
LFI 18-12 [1]
Register 18-17 [1]
LFI_CON **18-17 [1]**
Offset addresses 18-17 [1]
Restrictions 18-13 [1]
Link
Instruction 2-58 [1]
LMB 18-1 [1], 18-25 [1]
Bus agent priorities 18-25 [1]
Error capture 18-5 [1]
Load
Bit 2-63 [1]
Instructions 2-61 [1]
Load Effective Address 2-56 [1]
Logic operations 2-43 [1]

Loop instructions 2-60 [1]
Lower Context 2-7 [1], 2-15 [1]

M

MCHK 17-90 [1]
Functionality 17-90 [1]
Registers 17-91 [1]
Address range 17-95 [1]
IR **17-92 [1]**
RR **17-93 [1]**
WR **17-94 [1]**
MEM 16-10 [1]
Memory checker module, see “MCHK” 17-90 [1]
Memory management unit 10-1 [1]–10-19 [1]
Registers 10-12 [1]
Address range 10-19 [1]
MMU_ASI **10-15 [1]**
MMU_CON **10-13 [1]**
MMU_TFA **10-18 [1]**
MMU_TPA **10-16 [1]**
MMU_TPX **10-18 [1]**
MMU_TVA **10-15 [1]**
Offset addresses 10-13 [1]
Overview 10-13 [1]
Memory protection system 12-1 [1]–12-20 [1]
Configuration example 12-17 [1]
Memory access checking 12-18 [1]
Overview 12-1 [1]
Registers
Address Range 12-20 [1]
Control by PSW bits/bit fields 12-7 [1]
for code memory protection 12-14 [1]
for data memory protection 12-11 [1]
Offset addresses 12-4 [1]
Overview 12-2 [1]
MFCR 2-14 [1]
MLI
Applications 27-2 [2]
Communication principles 27-6 [2]
DMA requests 27-106 [2]
General description 27-7 [2]
Handshake timing 27-13 [2]
Interrupts 27-47 [2]
Naming conventions 27-5 [2]
Reading process 27-46 [2]
Receiver 27-35 [2]
Error handling 27-43 [2]
I/O control 27-44 [2]
Operation modes 27-35 [2]
Registers
Address ranges 27-119 [2]
AER **27-95 [2]**
ARR **27-96 [2]**
GINTR **27-94 [2]**
Offset addresses 27-60 [2]
OICR **27-81 [2]**
Overview 27-59 [2]
RADDR **27-79 [2]**
RCR **27-74 [2]**
RDATAR **27-79 [2]**
RIER **27-90 [2]**
RINPR **27-93 [2]**
RISR **27-92 [2]**
RPxBAR **27-77 [2]**
RPxSTATR **27-78 [2]**
SCR **27-80 [2]**
TCBAR **27-73 [2]**
TCMDR **27-67 [2]**
TCR **27-62 [2]**
TDRAR **27-72 [2]**
TIER **27-86 [2]**
TINPR **27-88 [2]**
TISR **27-87 [2]**
TPxAOFR **27-71 [2]**
TPxBAR **27-72 [2]**
TPxDATAR **27-71 [2]**
TPxSTATR **27-66 [2]**
TRSTATR **27-69 [2]**
TSTATR **27-64 [2]**
Startup 27-15 [2]

Timings 27-49 [2]
Transaction flow diagrams
Copy base address 27-54 [2]
Transmitter 27-18 [2]
Errors 27-31 [2]
I/O control 27-32 [2]
Operation modes 27-18 [2]
Parity 27-31 [2]
Transfer mode selection 27-28 [2]
Transmission format 27-20 [2]
Transmission modes 27-21 [2]
MMU 16-8 [1]
MMU, see “Memory management unit”
Module clock generation 3-22 [1]
Move 2-40 [1]
MPN 16-9 [1]
MPP 16-9 [1]
MPR 16-8 [1]
MPW 16-9 [1]
MPX 16-9 [1]
MTCR 2-14 [1]
MUL 2-41 [1]
MULS 2-41 [1]
MULS.U 2-41 [1]
MultiCAN 1-27 [1]
Multiply
Add 2-41 [1], 2-49 [1], 2-50 [1]

N

NEST 16-13 [1]
Nesting 16-4 [1]
NMI 16-14 [1], 16-19 [1]
from external 16-21 [1]
from PLL 16-21 [1]
from watchdog timer 16-21 [1]
Input pin 16-21 [1]
Non-Maskable Interrupt 16-4 [1]
Status register NMISR 16-19 [1], 16-20 [1]
NOP 2-67 [1]
No-operation 2-67 [1]
Normalization 2-43 [1]

O

OCDS 21-1 [1]–21-86 [1]
Address ranges 21-86 [1]
Break conditions 21-6 [1]
Features 21-2 [1]
Level 1 21-1 [1], 21-4 [1]
Level 2 21-1 [1]
Module implementation
Input/output function selection 21-69 [1]
Multi core break switch 21-33 [1]
Break bus switch 21-37 [1]
Features 21-33 [1]
Registers
MCDBBS **21-38 [1]**
MCDBBSS **21-39 [1]**
MCDSSG **21-42 [1]**
MCDSSGC **21-44 [1]**
OCDS System control unit (OSCU) 21-21 [1]
Registers 21-21 [1]
OSCU System control unit (OSCU)
Enable and reset 21-25 [1]
Product chip 21-4 [1]
Registers
Address ranges 21-86 [1]
System block diagram 21-2 [1]
System control unit (OSCU)
Registers
OCNTRL **21-21 [1]**
OEC **21-24 [1]**
OJCONF **21-24 [1]**
OSTATE **21-22 [1]**
OCDS, see “On-Chip Debug Support”
On-chip debug support
Registers
CREVT **2-35 [1]**
DBGSR **2-32 [1]**
EXEVT **2-34 [1]**
SWEVT **2-36 [1]**
TR0EVT **2-37 [1]**
TR1EVT **2-37 [1]**

On-chip memories 7-1 [1]–7-6 [1]
On-chip peripherals overview 1-11 [1]
OPD 16-10 [1]
Overflow 2-49 [1], 16-4 [1]
OVF 16-14 [1]

P

Packed
  Arithmetic instructions 2-47 [1]
  Bytes 2-53 [1]
  Values 2-48 [1]
Parallel ports
  Kernel registers 13-4 [1]
    Px_ALTSELn **13-11 [1]**
    Px_DIR **13-7 [1]**
    Px_OD **13-8 [1]**
    Px_PUDEN **13-10 [1]**
    Px_PUDSEL **13-9 [1]**
PC
  Program Counter 2-14 [1]
  Return 16-6 [1]
Pending
  Interrupt Priority Number (PIPN) 2-27 [1]
Pin configuration 1-32 [1]
Pin diagram 1-33 [1]
PIPN 2-27 [1]
PLL **3-10 [1]**
  Features 3-10 [1]
  Functionality 3-11 [1]
  Lock detection 3-19 [1]
  Loss-of-lock 3-19 [1]
  PLL_CLC **3-14 [1]**
  Setup after reset 3-18 [1]
  Startup 3-19 [1]
  Switching parameters 3-18 [1]
PMI
  Registers
    PMI_CON0 **8-7 [1]**
    PMI_CON1 **8-8 [1]**
    PMI_CON2 **8-9 [1]**
PMI, see “Program memory interface”
Port 0 13-12 [1]
Port 1 13-17 [1]
Port 2 13-22 [1]
Port 3 13-27 [1]
Port 4 13-32 [1]
Ports 13-1 [1]–13-35 [1]
  General port structure 13-2 [1]
  Kernel registers
    Offset addresses 13-4 [1]
  Port 0 13-12 [1]
    I/O functions 13-13 [1]
  Port 1 13-17 [1]
    I/O functions 13-18 [1]
  Port 2 13-22 [1]
    I/O functions 13-23 [1]
  Port 3 13-27 [1]
    I/O functions 13-28 [1]
  Port 4 13-32 [1]
    I/O functions 13-33 [1]
Power Management 6-1 [1]–6-10 [1]
  Mode description 6-6 [1]
    Deep sleep mode 6-7 [1]
    Idle mode 6-6 [1]
    Sleep mode 6-6 [1]
  Overview 6-1 [1]
  Registers 6-3 [1]
    Address range 6-3 [1]
    Offset addresses 6-3 [1]
    Overview 6-3 [1]
    PMG_CON **6-3 [1]**
    PMG_CSR **6-5 [1]**
Power management
  Mode definitions 6-2 [1]
PPN
  Physical Page Number 10-5 [1]
Priorities of bus agents
  FPI 18-26 [1]
  LMB 18-25 [1]
PRIV 16-8 [1]
Program
  Counter 2-14 [1]
Program memory interface 8-1 [1]–8-9 [1]
  Block diagram 8-2 [1]
  Features 8-1 [1]

Instruction cache 8-4 [1]
Registers 8-6 [1]
Offset addresses 8-6 [1]
Overview 8-6 [1]
Scratch-pad code RAM 8-2 [1]
PS
Program Status 2-9 [1]
PSE 16-13 [1]
PSW 2-39 [1]
Program Status Word 2-7 [1]
PTE
Page Table Entry 2-10 [1]

R

RA
Return Address 2-14 [1]
RE
Read Enable 10-5 [1]
Redundant zeros 2-43 [1]
Register
Data 2-14 [1]
Extended-size 2-14 [1]
Floating point 2-14 [1]
System global 2-14 [1]
Register overview 22-1 [1]
Reset
Debug system reset 5-9 [1]
External hardware reset 5-6 [1]
Power-on reset 5-6 [1]
Registers
RST_REQ **5-5 [1]**
RST_SR **5-3 [1]**
Software reset 5-7 [1]
WDT reset 5-7 [1]
Reset and boot operation 5-1 [1]–5-15 [1]
Reset operation
Overview 5-1 [1]
Registers
Address range 5-2 [1]
Offset addresses 5-2 [1]
Overview 5-2 [1]
RST_REQ **5-4 [1]**
RST_SR **5-2 [1]**
States after reset 5-9 [1]
Reset Overflow Flags 2-14 [1]
Return
Address 2-14 [1]
Instruction 2-67 [1]
Rounding 2-49 [1]

S

Saturate Instructions 2-42 [1]
Saturation 2-49 [1]
SBCU
Registers 18-27 [1]
Offset addresses 18-27 [1]
Overview 18-27 [1]
RBCU_DBADR1 **18-37 [1]**
SBCU_CON **18-28 [1]**
SBCU_DBADR2 **18-37 [1]**
SBCU_DBADRT **18-35 [1]**
SBCU_DBBOS **18-36 [1]**
SBCU_DBBOST **18-33 [1]**
SBCU_DBCNTL **18-39 [1]**
SBCU_DBGNTT **18-35 [1]**
SBCU_DBGRNT **18-38 [1]**
SBCU_EADD **18-32 [1]**
SBCU_ECON **18-31 [1]**
SBCU_EDAT **18-32 [1]**
SBCU_SRC **18-42 [1]**
Scaled
Offset instruction 2-63 [1]
Scaling 2-48 [1]
SCU
Address range 4-22 [1]
CSCOMB control 4-8 [1]
DMA request signal selection 4-9 [1]
Faulty SRAM fusebox 4-6 [1]
Miscellaneous registers
CHIPID **4-17 [1]**
MANID **4-16 [1]**
RTID **4-18 [1]**
SCU_CON **4-12 [1]**
Overview 4-1 [1]
Parity error control 4-2 [1]
Registers

Address range 4-22 [1]
Offset addresses 4-19 [1]
Overview 4-19 [1]
SCU_STAT **4-15 [1]**
SCU, see “System Control Unit”
SDRAM interface 14-88 [1]
Serial interfaces 1-12 [1]
Async./sync. serial interface 1-12 [1]
High-speed sync. serial interface 1-15 [1]
IIC 1-17 [1]
MLI 1-21 [1]
USB 1-19 [1]
Shift instructions 2-44 [1]
SMT
Software Managed Tasks 2-7 [1]
SOVF 16-14 [1]
SP
Stack Pointer 2-14 [1]
SPRAM 8-2 [1]
SRN
Service Request Nodes 2-9 [1]
SRPN
Service Request Priority Number 2-9 [1]
SSC
Address ranges 24-75 [2]
Baud rate generation 24-19 [2], 24-25 [2]
Block diagram 24-5 [2]
Chip select generation 24-23 [2]
DMA request outputs 24-75 [2]
Error detection 24-26 [2]
FIFO operation
Receive FIFO 24-15 [2]
Transmit FIFO 24-13 [2]
Transparent Mode 24-17 [2]
Full-duplex operation 24-7 [2]
Half-duplex operation 24-10 [2]
Interrupts 24-26 [2]
Module implementation 24-45 [2]–24-75 [2]
DMA request outputs 24-75 [2]
Interrupt registers 24-74 [2]
Module clock control 24-48 [2]
Port input select 24-52 [2]
Registers 24-28 [2]–24-39 [2]
Address ranges 24-75 [2]
BR **24-38 [2]**
CON **24-31 [2]**
EFM **24-34 [2]**
FSTAT **24-44 [2]**
Offset addresses 24-28 [2]
Overview 24-28 [2]
PISEL **24-29 [2]**, **24-55 [2]**
RB **24-39 [2]**
RXFCON **24-40 [2]**
SSOC **24-36 [2]**
SSOTC **24-37 [2]**
STAT **24-33 [2]**
TB **24-39 [2]**
TXFCON **24-42 [2]**
Slave select input operation 24-21 [2]
Stack Pointer 2-14 [1]
Status flags 2-39 [1]
Sticky overflow 16-4 [1]
STM, see “System Timer”
Store
Bit 2-63 [1]
Instructions 2-61 [1]
Supervisor 2-7 [1]
Synchronization primitives 2-65 [1]
Synchronous trap 16-4 [1]
SYS 16-14 [1]
System
Call 2-65 [1], 16-4 [1]
Global registers 2-14 [1]
Instructions 2-65 [1]
System control unit 4-1 [1]–4-22 [1]
System Global Registers 2-8 [1]
System timer 19-1 [1]–19-19 [1]
Block diagram 19-2 [1]
Compare register operation 19-4 [1]
Interrupt control 19-5 [1]
Overview 19-1 [1]
Registers

Address range 19-19 [1]
CAP **19-8 [1]**
CMCON **19-10 [1]**
CMPx **19-9 [1]**
ICR **19-12 [1]**
ISRR **19-14 [1]**
Offset addresses 19-6 [1]
Overview 19-6 [1]
TIM0 **19-7 [1]**
TIM1 **19-7 [1]**
TIM2 **19-7 [1]**
TIM3 **19-7 [1]**
TIM4 **19-8 [1]**
TIM5 **19-8 [1]**
TIM6 **19-8 [1]**
Resolutions and ranges 19-3 [1]

T

Table indexes 2-14 [1]
Timer units
General purpose timer unit 1-23 [1]
TIN 16-2 [1]
Trap Identification Number 2-10 [1]
TLB 10-5 [1], 16-8 [1]
Trap 16-2 [1], 16-8 [1], 16-15 [1]
Default state 16-7 [1]
Handler vector 16-5 [1]
Vector table 16-6 [1]
Trap system 16-1 [1]–16-21 [1]
Overview 16-1 [1]
Service routine 16-18 [1]
Trap vector table 16-17 [1]
Traps 2-14 [1]
Asynchronous 16-5 [1]
Synchronous 16-4 [1]
TTE 10-5 [1]
fields 10-5 [1]

U

Unconditional branch instructions 2-58 [1]
UOPC 16-10 [1]
Upper Context 2-7 [1], 2-15 [1]
USB
Access Control 26-6 [2]
Address ranges 26-75 [2]
Assembly buffers 26-6 [2]
Control transfers 26-14 [2]
DMA request outputs 26-74 [2]
Features 26-2 [2]
Initialization 26-11 [2]
Memory 26-6 [2]
MMUs of USB 26-8 [2]
Module implementation
26-66 [2]–26-74 [2]
DMA request outputs 26-74 [2]
Interrupt registers 26-74 [2]
Module clock control 26-68 [2]
Peripheral input select 26-69 [2]
Registers 26-21 [2]–26-64 [2]
Address ranges 26-75 [2]
CLC **26-68 [2]**
CNFR **26-34 [2]**
CPLPR **26-48 [2]**
DATA16 **26-50 [2]**
DATA32 **26-49 [2]**
DATA8 **26-50 [2]**
DCR **26-27 [2]**
DIER **26-54 [2]**
DINP **26-53 [2]**
DIRR **26-56 [2]**
DIRST **26-58 [2]**
DSR **26-29 [2]**
EPBCn **26-39 [2]**
EPCPn **26-38 [2]**
EPDIR **26-40 [2]**
EPDSR **26-41 [2]**
EPICn **26-60 [2]**
EPIRn **26-63 [2]**
EPIRSTn **26-64 [2]**
EPSSR **26-33 [2]**
EPSTL **26-32 [2]**
EPUPn **26-37 [2]**
EPVLD **26-43 [2]**
EVSR **26-44 [2]**
FCON **26-45 [2]**
FNR **26-34 [2]**

Offset addresses 26-21 [2]
Overview 26-21 [2]
PISEL **26-26 [2]**
SUTH **26-35 [2]**
SUTL **26-35 [2]**
ZLPEN **26-42 [2]**
ZLPSR **26-42 [2]**
Transfer modes 26-10 [2]
User-0 2-7 [1]
User-1 2-7 [1]

V

VAF 16-8 [1]
VAP 16-8 [1]
VPN
Virtual Page Number 2-10 [1], 10-5 [1]

W

Watchdog timer 20-1 [1]–20-34 [1]
During power-saving modes 20-17 [1]
Endinit function 20-3 [1]
Features 20-2 [1]
Functional description 20-5 [1]
in OCDS suspend mode 20-17 [1]
Modes of operation 20-7 [1]
Disable mode 20-8 [1], 20-15 [1]
Normal mode 20-8 [1], 20-14 [1]
Prewarning mode 20-9 [1], 20-16 [1]
Time-out mode 20-8 [1], 20-13 [1]
Modify access to WDT_CON0 20-11 [1]
Monitoring diagram 20-26 [1]
Operation sequence example 20-5 [1]
Overview 20-1 [1]
Period calculation 20-18 [1]
Period in power-saving modes 20-20 [1]
Registers 20-28 [1]
Offset addresses 20-28 [1]
WDT_CON0 **20-29 [1]**
WDT_CON1 **20-31 [1]**
WDT_SR **20-32 [1]**
Service sequence diagram 20-25 [1]
Servicing 20-23 [1]
System initialization 20-22 [1]
Time-out period 20-18 [1]
WDT, see “Watchdog timer”
WE
Write Enable 10-5 [1]

X

XE
Execute Enable 10-5 [1]

Register Index

This User's Manual consists of two volumes, "System Units" [1] and "Peripheral Units" [2]. For your convenience, this register index (and also the table of contents and the keyword index) lists both volumes, so you can immediately find the reference to the desired section in the corresponding document ([1] or [2]).

A

A0 22-82 [1]
A1 22-82 [1]
A10 22-82 [1]
A11 22-82 [1]
A12 22-82 [1]
A13 22-82 [1]
A14 22-82 [1]
A15 22-83 [1]
A2 22-82 [1]
A3 22-82 [1]
A4 22-82 [1]
A5 22-82 [1]
A6 22-82 [1]
A7 22-82 [1]
A8 22-82 [1]
A9 22-82 [1]
ASC module registers 23-29 [2]
ASC0_BG 22-59 [1], 23-34 [2]
ASC0_CLC 22-59 [1], 23-46 [2]
ASC0_CON 22-59 [1], 23-30 [2]
ASC0_ESRC 22-60 [1], 23-58 [2]
ASC0_FDV 22-59 [1], 23-35 [2]
ASC0_FSTAT 22-59 [1], 23-41 [2]
ASC0_ID 22-59 [1]
ASC0_PISEL 22-59 [1], 23-30 [2]
ASC0_PMW 23-36 [2]
ASC0_RBUF 22-59 [1], 23-37 [2]
ASC0_RSRC 22-60 [1], 23-58 [2]
ASC0_RXFCON 22-59 [1], 23-38 [2]
ASC0_TBSRC 22-60 [1], 23-58 [2]
ASC0_TBUF 22-59 [1], 23-36 [2]
ASC0_TSRC 22-60 [1], 23-58 [2]
ASC0_TXFCON 22-59 [1], 23-40 [2]
ASC0_WHBCON 22-59 [1], 23-33 [2]
ASC1_BG 22-60 [1], 23-34 [2]
ASC1_CLC 22-60 [1], 23-46 [2]
ASC1_CON 22-60 [1], 23-30 [2]
ASC1_ESRC 22-61 [1], 23-58 [2]
ASC1_FDV 22-60 [1], 23-35 [2]
ASC1_FSTAT 22-61 [1], 23-41 [2]
ASC1_ID 22-60 [1]
ASC1_PISEL 22-60 [1], 23-30 [2], 23-48 [2]
ASC1_PMW 22-60 [1], 23-36 [2]
ASC1_RBUF 22-60 [1], 23-37 [2]
ASC1_RSRC 22-61 [1], 23-58 [2]
ASC1_RXFCON 22-61 [1], 23-38 [2]
ASC1_TBSRC 22-61 [1], 23-58 [2]
ASC1_TBUF 22-60 [1], 23-36 [2]
ASC1_TSRC 22-61 [1], 23-58 [2]
ASC1_TXFCON 22-61 [1], 23-40 [2]
ASC1_WHBCON 22-61 [1], 23-33 [2]
ASC2_BG 22-62 [1], 23-34 [2]
ASC2_CLC 22-61 [1], 23-46 [2]
ASC2_CON 22-61 [1], 23-30 [2]
ASC2_ESRC 22-63 [1], 23-58 [2]
ASC2_FDV 22-62 [1], 23-35 [2]
ASC2_FSTAT 22-62 [1], 23-41 [2]
ASC2_ID 22-61 [1]
ASC2_PISEL 22-61 [1], 23-30 [2], 23-48 [2]
ASC2_PMW 22-62 [1], 23-36 [2]
ASC2_RBUF 22-62 [1], 23-37 [2]
ASC2_RSRC 22-62 [1], 23-58 [2]
ASC2_RXFCON 22-62 [1], 23-38 [2]
ASC2_TBSRC 22-63 [1], 23-58 [2]
ASC2_TBUF 22-62 [1], 23-36 [2]
ASC2_TSRC 22-62 [1], 23-58 [2]

ASC2_TXFCON 22-62 [1], 23-40 [2]
ASC2_WHBCON 22-62 [1], 23-33 [2]

B

BIV 2-29 [1], 22-80 [1]
BTV 2-30 [1], 22-80 [1]

C

CAN module registers 30-41 [2]
CAN_CLC 22-40 [1], 30-91 [2]
CAN_FDR 22-40 [1], 30-92 [2]
CAN_ID 22-40 [1]
CAN_LIST0 22-41 [1], 30-48 [2]
CAN_LIST1 22-41 [1], 30-48 [2]
CAN_LIST2 22-41 [1], 30-48 [2]
CAN_LIST3 22-41 [1], 30-48 [2]
CAN_LIST4 22-41 [1], 30-48 [2]
CAN_LIST5 22-41 [1], 30-48 [2]
CAN_LIST6 22-41 [1], 30-48 [2]
CAN_LIST7 22-41 [1], 30-48 [2]
CAN_MCR 22-43 [1], 30-45 [2]
CAN_MITR 22-43 [1], 30-46 [2]
CAN_MOAMR0 22-45 [1]
CAN_MOAMR1 22-46 [1]
CAN_MOAMR126 22-48 [1]
CAN_MOAMR127 22-48 [1]
CAN_MOAMR2 22-46 [1]
CAN_MOAMRn 22-47 [1], 30-79 [2]
CAN_MOAR0 22-45 [1]
CAN_MOAR1 22-46 [1]
CAN_MOAR126 22-48 [1]
CAN_MOAR127 22-48 [1]
CAN_MOAR2 22-46 [1]
CAN_MOARn 22-47 [1], 30-80 [2]
CAN_MOCTR0 22-45 [1]
CAN_MOCTR1 22-46 [1]
CAN_MOCTR126 22-48 [1]
CAN_MOCTR127 22-49 [1]
CAN_MOCTR2 22-47 [1]
CAN_MOCTRn 22-47 [1], 30-68 [2]
CAN_MODATA00 22-45 [1]
CAN_MODATA04 22-45 [1]
CAN_MODATA10 22-46 [1]
CAN_MODATA1260 22-48 [1]
CAN_MODATA1264 22-48 [1]
CAN_MODATA1270 22-48 [1]
CAN_MODATA1274 22-49 [1]
CAN_MODATA14 22-46 [1]
CAN_MODATA20 22-46 [1]
CAN_MODATA24 22-46 [1]
CAN_MODATAHn 30-84 [2]
CAN_MODATALn 30-83 [2]
CAN_MODATAn0 22-47 [1]
CAN_MODATAn4 22-47 [1]
CAN_MOFCR0 22-45 [1]
CAN_MOFCR1 22-45 [1]
CAN_MOFCR126 22-47 [1]
CAN_MOFCR127 22-48 [1]
CAN_MOFCR2 22-46 [1]
CAN_MOFCRn 22-47 [1], 30-74 [2]
CAN_MOFGPR0 22-45 [1]
CAN_MOFGPR1 22-46 [1]
CAN_MOFGPR126 22-47 [1]
CAN_MOFGPR127 22-48 [1]
CAN_MOFGPR2 22-46 [1]
CAN_MOFGPRn 22-47 [1], 30-78 [2]
CAN_MOIPR0 22-45 [1]
CAN_MOIPR1 22-46 [1]
CAN_MOIPR126 22-48 [1]
CAN_MOIPR127 22-48 [1]
CAN_MOIPR2 22-46 [1]
CAN_MOIPRn 22-47 [1], 30-73 [2]
CAN_MSID0 22-42 [1], 30-49 [2]
CAN_MSID1 22-42 [1], 30-49 [2]
CAN_MSID2 22-42 [1], 30-49 [2]
CAN_MSID3 22-42 [1], 30-49 [2]
CAN_MSID4 22-42 [1], 30-49 [2]
CAN_MSID5 22-42 [1], 30-49 [2]
CAN_MSID6 22-42 [1], 30-49 [2]
CAN_MSID7 22-42 [1], 30-49 [2]
CAN_MSIMASK 22-42 [1], 30-50 [2]
CAN_MSPND0 22-41 [1], 30-49 [2]
CAN_MSPND1 22-41 [1], 30-49 [2]
CAN_MSPND2 22-41 [1], 30-49 [2]
CAN_MSPND3 22-42 [1], 30-49 [2]
CAN_MSPND4 22-42 [1], 30-49 [2]

CAN_MSPND5 22-42 [1], 30-49 [2]
CAN_MSPND6 22-42 [1], 30-49 [2]
CAN_MSPND7 22-42 [1], 30-49 [2]
CAN_NBTR0 22-43 [1], 30-61 [2]
CAN_NBTR1 22-44 [1], 30-61 [2]
CAN_NBTR2 22-44 [1], 30-61 [2]
CAN_NBTR3 22-45 [1], 30-61 [2]
CAN_NCR0 22-43 [1], 30-51 [2]
CAN_NCR1 22-43 [1], 30-51 [2]
CAN_NCR2 22-44 [1], 30-51 [2]
CAN_NCR3 22-44 [1], 30-51 [2]
CAN_NECNT0 22-43 [1], 30-63 [2]
CAN_NECNT1 22-44 [1], 30-63 [2]
CAN_NECNT2 22-44 [1], 30-63 [2]
CAN_NECNT3 22-45 [1], 30-63 [2]
CAN_NFCR0 22-43 [1], 30-64 [2]
CAN_NFCR1 22-44 [1], 30-64 [2]
CAN_NFCR2 22-44 [1], 30-64 [2]
CAN_NFCR3 22-45 [1], 30-64 [2]
CAN_NIPR0 22-43 [1], 30-59 [2]
CAN_NIPR1 22-43 [1], 30-59 [2]
CAN_NIPR2 22-44 [1], 30-59 [2]
CAN_NIPR3 22-44 [1], 30-59 [2]
CAN_NPCR0 22-43 [1], 30-60 [2]
CAN_NPCR1 22-43 [1], 30-60 [2]
CAN_NPCR2 22-44 [1], 30-60 [2]
CAN_NPCR3 22-45 [1], 30-60 [2]
CAN_NSR0 22-43 [1], 30-55 [2]
CAN_NSR1 22-43 [1], 30-55 [2]
CAN_NSR2 22-44 [1], 30-55 [2]
CAN_NSR3 22-44 [1], 30-55 [2]
CAN_PANCTR 22-42 [1], 30-42 [2]
CAN_SRC0 22-41 [1]
CAN_SRC1 22-41 [1]
CAN_SRC10 22-40 [1]
CAN_SRC11 22-40 [1]
CAN_SRC12 22-40 [1]
CAN_SRC13 22-40 [1]
CAN_SRC14 22-40 [1]
CAN_SRC15 22-40 [1]
CAN_SRC2 22-41 [1]
CAN_SRC3 22-41 [1]
CAN_SRC4 22-41 [1]
CAN_SRC5 22-40 [1]
CAN_SRC6 22-40 [1]
CAN_SRC7 22-40 [1]
CAN_SRC8 22-40 [1]
CAN_SRC9 22-40 [1]
CAN_SRCm 30-101 [2]
CBS_COMDATA 21-54 [1], 22-17 [1]
CBS_IOSR 21-52 [1], 22-17 [1]
CBS_JDPID 22-17 [1]
CBS_MCDBBS 21-38 [1], 22-17 [1]
CBS_MCDBBSS 21-39 [1], 22-17 [1]
CBS_MCDSSG 21-42 [1], 22-17 [1]
CBS_MCDSSGC 21-44 [1], 22-17 [1]
CBS_OCNTRL 21-21 [1], 22-17 [1]
CBS_OEC 21-24 [1], 22-17 [1]
CBS_OSTATE 21-22 [1], 22-17 [1]
CBS_SRC 21-31 [1], 22-18 [1]
CCU6 module registers 29-36 [2]
CCU60_63R 22-28 [1]
CCU60_63SR 22-28 [1]
CCU60_CC60R 22-27 [1]
CCU60_CC60SR 22-27 [1]
CCU60_CC61R 22-27 [1]
CCU60_CC61SR 22-27 [1]
CCU60_CC62R 22-27 [1]
CCU60_CC62SR 22-27 [1]
CCU60_CC63R 29-60 [2]
CCU60_CC63SR 29-61 [2]
CCU60_CC6xR 29-54 [2]
CCU60_CC6xSR 29-55 [2]
CCU60_CLC 22-27 [1], 29-96 [2]
CCU60_CMPMODIF 22-28 [1], 29-43 [2]
CCU60_CMPSTAT 22-28 [1], 29-41 [2]
CCU60_FDR 22-27 [1], 29-97 [2]
CCU60_ID 22-27 [1]
CCU60_IEN 22-29 [1], 29-85 [2]
CCU60_INP 22-29 [1], 29-89 [2]
CCU60_IS 22-28 [1], 29-78 [2]
CCU60_ISR 22-29 [1], 29-83 [2]
CCU60_ISS 22-29 [1], 29-81 [2]
CCU60_MCMCTR 22-28 [1], 29-72 [2]
CCU60_MCMOUT 22-28 [1], 29-69 [2]
CCU60_MCMOUTS 22-28 [1], 29-68 [2]

CCU60_MODCTR 22-28 [1], 29-62 [2]
CCU60_PISEL0 22-27 [1], 29-38 [2]
CCU60_PISEL2 22-27 [1], 29-40 [2]
CCU60_PSLR 22-28 [1], 29-66 [2]
CCU60_SRC0 22-29 [1]
CCU60_SRC1 22-29 [1]
CCU60_SRC2 22-29 [1]
CCU60_SRC3 22-29 [1]
CCU60_SRCx 29-110 [2]
CCU60_T12 22-27 [1], 29-52 [2]
CCU60_T12DTC 22-27 [1], 29-56 [2]
CCU60_T12MSEL 22-28 [1], 29-74 [2]
CCU60_T12PR 22-27 [1], 29-53 [2]
CCU60_T13 22-27 [1], 29-58 [2]
CCU60_T13PR 22-28 [1], 29-59 [2]
CCU60_TCTR0 22-28 [1], 29-44 [2]
CCU60_TCTR2 22-28 [1], 29-47 [2]
CCU60_TCTR4 22-28 [1], 29-50 [2]
CCU60_TRPCTR 22-28 [1], 29-64 [2]
CCU61_63R 22-30 [1]
CCU61_63SR 22-31 [1]
CCU61_CC60R 22-30 [1]
CCU61_CC60SR 22-30 [1]
CCU61_CC61R 22-30 [1]
CCU61_CC61SR 22-30 [1]
CCU61_CC62R 22-30 [1]
CCU61_CC62SR 22-30 [1]
CCU61_CC63R 29-60 [2]
CCU61_CC63SR 29-61 [2]
CCU61_CC6xR 29-54 [2]
CCU61_CC6xSR 29-55 [2]
CCU61_CMPMODIF 22-31 [1], 29-43 [2]
CCU61_CMPSTAT 22-31 [1], 29-41 [2]
CCU61_ID 22-30 [1]
CCU61_IEN 22-32 [1], 29-85 [2]
CCU61_INP 22-32 [1], 29-89 [2]
CCU61_IS 22-31 [1], 29-78 [2]
CCU61_ISR 22-32 [1], 29-83 [2]
CCU61_ISS 22-31 [1], 29-81 [2]
CCU61_MCMCTR 22-31 [1], 29-72 [2]
CCU61_MCMOUT 22-31 [1], 29-69 [2]
CCU61_MCMOUTS 22-31 [1], 29-68 [2]
CCU61_MODCTR 22-31 [1], 29-62 [2]
CCU61_PISEL0 22-30 [1], 29-38 [2]
CCU61_PISEL2 22-30 [1], 29-40 [2]
CCU61_PSLR 22-31 [1], 29-66 [2]
CCU61_SRC0 22-32 [1]
CCU61_SRC1 22-32 [1]
CCU61_SRC2 22-32 [1]
CCU61_SRC3 22-32 [1]
CCU61_SRCx 29-110 [2]
CCU61_T12 22-30 [1], 29-52 [2]
CCU61_T12DTC 22-30 [1], 29-56 [2]
CCU61_T12MSEL 22-31 [1], 29-74 [2]
CCU61_T12PR 22-30 [1], 29-53 [2]
CCU61_T13 22-30 [1], 29-58 [2]
CCU61_T13PR 22-30 [1], 29-59 [2]
CCU61_TCTR0 22-31 [1], 29-44 [2]
CCU61_TCTR2 22-31 [1], 29-47 [2]
CCU61_TCTR4 22-31 [1], 29-50 [2]
CCU61_TRPCTR 22-31 [1], 29-64 [2]
CHIPID 4-17 [1], 22-11 [1]
Clock registers 3-5 [1]
CPM0 12-15 [1], 22-79 [1]
CPM1 12-15 [1], 22-79 [1]
CPR0_0L 12-14 [1], 22-78 [1]
CPR0_0U 12-14 [1], 22-78 [1]
CPR0_1L 12-14 [1], 22-78 [1]
CPR0_1U 12-14 [1], 22-78 [1]
CPR1_0L 12-14 [1], 22-78 [1]
CPR1_0U 12-14 [1], 22-78 [1]
CPR1_1L 12-14 [1], 22-78 [1]
CPR1_1U 12-14 [1], 22-78 [1]
CPS_ID 22-75 [1]
CPU_ID 22-80 [1]
CPU_SBSRC 2-80 [1], 22-75 [1]
CPU_SRC0 2-82 [1], 22-75 [1]
CPU_SRC1 2-82 [1], 22-75 [1]
CPU_SRC2 2-82 [1], 22-75 [1]
CPU_SRC3 2-82 [1], 22-75 [1]
CREVT 2-35 [1], 22-79 [1]
CSCACTL 11-11 [1], 22-89 [1]
CSCADIN 11-12 [1], 22-89 [1]
CSCADOUT 11-13 [1], 22-89 [1]

D

D0 22-81 [1]
D1 22-81 [1]
D10 22-81 [1]
D11 22-81 [1]
D12 22-81 [1]
D13 22-82 [1]
D14 22-82 [1]
D15 22-82 [1]
D2 22-81 [1]
D3 22-81 [1]
D4 22-81 [1]
D5 22-81 [1]
D6 22-81 [1]
D7 22-81 [1]
D8 22-81 [1]
D9 22-81 [1]
DBGSR 2-32 [1], 22-79 [1]
DCX 2-38 [1], 22-80 [1]
DMA controller registers 17-34 [1]
DMA_ADRCR00 22-34 [1]
DMA_ADRCR01 22-35 [1]
DMA_ADRCR02 22-35 [1]
DMA_ADRCR03 22-36 [1]
DMA_ADRCR04 22-36 [1]
DMA_ADRCR05 22-37 [1]
DMA_ADRCR06 22-37 [1]
DMA_ADRCR07 22-38 [1]
DMA_ADRCR0n 17-66 [1]
DMA_CHCR00 22-34 [1]
DMA_CHCR01 22-35 [1]
DMA_CHCR02 22-35 [1]
DMA_CHCR03 22-36 [1]
DMA_CHCR04 22-36 [1]
DMA_CHCR05 22-36 [1]
DMA_CHCR06 22-37 [1]
DMA_CHCR07 22-37 [1]
DMA_CHCR0n 17-59 [1]
DMA_CHICR00 22-34 [1]
DMA_CHICR01 22-35 [1]
DMA_CHICR02 22-35 [1]
DMA_CHICR03 22-36 [1]
DMA_CHICR04 22-36 [1]
DMA_CHICR05 22-37 [1]
DMA_CHICR06 22-37 [1]
DMA_CHICR07 22-37 [1]
DMA_CHICR0n 17-64 [1]
DMA_CHRSTR 17-41 [1], 22-33 [1]
DMA_CHSR00 22-34 [1]
DMA_CHSR01 22-35 [1]
DMA_CHSR02 22-35 [1]
DMA_CHSR03 22-35 [1]
DMA_CHSR04 22-36 [1]
DMA_CHSR05 22-36 [1]
DMA_CHSR06 22-37 [1]
DMA_CHSR07 22-37 [1]
DMA_CHSR0n 17-63 [1]
DMA_CLC 17-84 [1], 22-33 [1]
DMA_CLRE 17-49 [1], 22-33 [1]
DMA_DADR00 22-34 [1]
DMA_DADR01 22-35 [1]
DMA_DADR02 22-35 [1]
DMA_DADR03 22-36 [1]
DMA_DADR04 22-36 [1]
DMA_DADR05 22-37 [1]
DMA_DADR06 22-37 [1]
DMA_DADR07 22-38 [1]
DMA_DADR0n 17-71 [1]
DMA_EER 17-45 [1], 22-33 [1]
DMA_ERRSR 17-47 [1], 22-33 [1]
DMA_GINTR 17-40 [1], 22-33 [1]
DMA_HTREQ 17-44 [1], 22-33 [1]
DMA_ID 22-33 [1]
DMA_INTCR 17-53 [1], 22-34 [1]
DMA_INTSR 17-51 [1], 22-34 [1]
DMA_ME0AENR 17-56 [1], 22-34 [1]
DMA_ME0ARR 17-58 [1], 22-34 [1]
DMA_ME0PR 17-56 [1], 22-33 [1]
DMA_ME0R 17-55 [1], 22-33 [1]
DMA_MESR 17-54 [1], 22-33 [1]
DMA_MLI0SRC0 17-86 [1], 22-38 [1]
DMA_MLI0SRC1 17-86 [1], 22-38 [1]
DMA_MLI0SRC2 17-86 [1], 22-38 [1]
DMA_MLI0SRC3 17-86 [1], 22-38 [1]
DMA_MLI1SRC0 17-86 [1], 22-38 [1]

DMA_MLI1SRC1 17-86 [1], 22-38 [1]
DMA_OCDSR 17-36 [1], 22-34 [1]
DMA_SADR00 22-34 [1]
DMA_SADR01 22-35 [1]
DMA_SADR02 22-35 [1]
DMA_SADR03 22-36 [1]
DMA_SADR04 22-36 [1]
DMA_SADR05 22-37 [1]
DMA_SADR06 22-37 [1]
DMA_SADR07 22-38 [1]
DMA_SADR0n 17-70 [1]
DMA_SHADR00 22-34 [1]
DMA_SHADR01 22-35 [1]
DMA_SHADR02 22-35 [1]
DMA_SHADR03 22-36 [1]
DMA_SHADR04 22-36 [1]
DMA_SHADR05 22-37 [1]
DMA_SHADR06 22-37 [1]
DMA_SHADR07 22-38 [1]
DMA_SHADR0n 17-72 [1]
DMA_SRC0 17-85 [1], 22-39 [1]
DMA_SRC1 17-85 [1], 22-39 [1]
DMA_SRC2 17-85 [1], 22-39 [1]
DMA_SRC3 17-85 [1], 22-39 [1]
DMA_STREQ 17-43 [1], 22-33 [1]
DMA_SUSPMR 17-38 [1], 22-34 [1]
DMA_SYSSRC4 17-87 [1], 22-38 [1]
DMA_TOCTR 17-88 [1], 22-38 [1]
DMA_TRSR 17-42 [1], 22-33 [1]
DMA_WRPSR 17-52 [1], 22-34 [1]
DMARS 4-10 [1], 22-13 [1]
DMI module registers 9-3 [1]
DMI_ATR 9-6 [1], 22-90 [1]
DMI_CON 9-4 [1], 22-90 [1]
DMI_ID 22-90 [1]
DMI_STR 9-5 [1], 22-90 [1]
DMS 2-38 [1], 22-80 [1]
DMU module registers 11-8 [1]
DMU_ID 22-88 [1]
DPM0 12-12 [1], 22-78 [1]
DPM1 12-12 [1], 22-79 [1]
DPR0_0L 12-11 [1], 22-77 [1]
DPR0_0U 12-11 [1], 22-77 [1]
DPR0_1L 12-11 [1], 22-77 [1]
DPR0_1U 12-11 [1], 22-77 [1]
DPR0_2L 12-11 [1], 22-77 [1]
DPR0_2U 12-11 [1], 22-77 [1]
DPR0_3L 12-11 [1], 22-77 [1]
DPR0_3U 12-11 [1], 22-77 [1]
DPR1_0L 12-11 [1], 22-77 [1]
DPR1_0U 12-11 [1], 22-77 [1]
DPR1_1L 12-11 [1], 22-77 [1]
DPR1_1U 12-11 [1], 22-77 [1]
DPR1_2L 12-11 [1], 22-77 [1]
DPR1_2U 12-11 [1], 22-78 [1]
DPR1_3L 12-11 [1], 22-78 [1]
DPR1_3U 12-11 [1], 22-78 [1]

E

EBU module registers 14-109 [1]
EBU_ADDRSEL0 22-85 [1]
EBU_ADDRSEL1 22-85 [1]
EBU_ADDRSEL2 22-85 [1]
EBU_ADDRSEL3 22-85 [1]
EBU_ADDRSELx 14-113 [1]
EBU_BFCON 14-132 [1], 22-84 [1]
EBU_BUSAP0 22-86 [1]
EBU_BUSAP1 22-86 [1]
EBU_BUSAP2 22-86 [1]
EBU_BUSAP3 22-87 [1]
EBU_BUSAPx 14-119 [1]
EBU_BUSCON0 22-86 [1]
EBU_BUSCON1 22-86 [1]
EBU_BUSCON2 22-86 [1]
EBU_BUSCON3 22-86 [1]
EBU_BUSCONx 14-115 [1]
EBU_CLC 14-112 [1], 22-84 [1]
EBU_CON 14-130 [1], 22-84 [1]
EBU_EMUAS 14-122 [1], 22-87 [1]
EBU_EMUBAP 14-127 [1], 22-87 [1]
EBU_EMUBC 14-123 [1], 22-87 [1]
EBU_EMUOVL 14-129 [1], 22-87 [1]
EBU_ID 22-84 [1]
EBU_SDRMCON0 22-84 [1]
EBU_SDRMCON1 22-84 [1]
EBU_SDRMCONx 14-138 [1]

EBU_SDRMOD0 22-84 [1]
EBU_SDRMOD1 22-85 [1]
EBU_SDRMODx 14-141 [1]
EBU_SDRMREF0 22-84 [1]
EBU_SDRMREF1 22-84 [1]
EBU_SDRMREFx 14-136 [1]
EBU_SDRSTAT0 22-85 [1]
EBU_SDRSTAT1 22-85 [1]
EBU_SDRSTATx 14-142 [1]
EBU_USERCON 14-143 [1], 22-87 [1]
EICR0 15-38 [1], 22-12 [1]
EICR1 15-41 [1], 22-12 [1]
EIFR 15-44 [1], 22-12 [1]
EINT_SRC0 15-51 [1], 22-13 [1]
EINT_SRC1 15-51 [1], 22-13 [1]
EINT_SRC2 15-51 [1], 22-13 [1]
EINT_SRC3 15-51 [1], 22-12 [1]
Ethernet module registers 31-50 [2]
Ethernet_DRCMD 22-74 [1], 31-86 [2]
Ethernet_DRCONF 22-74 [1], 31-90 [2]
Ethernet_DRFFCR 22-74 [1], 31-85 [2]
Ethernet_DRFRDA 22-74 [1], 31-88 [2]
Ethernet_DRIMR 22-74 [1], 31-89 [2]
Ethernet_DRISFIFO 22-74 [1], 31-82 [2]
Ethernet_DRMOD 22-74 [1], 31-88 [2]
Ethernet_DRSRC 22-12 [1], 31-116 [2]
Ethernet_DTCMD 22-74 [1], 31-91 [2]
Ethernet_DTCONF 22-74 [1], 31-95 [2]
Ethernet_DTCONF3 22-74 [1], 31-95 [2]
Ethernet_DTFFCR 22-74 [1], 31-98 [2]
Ethernet_DTFTDA 22-74 [1], 31-93 [2]
Ethernet_DTIMR 22-74 [1], 31-94 [2]
Ethernet_DTISFIFO 22-74 [1], 31-96 [2]
Ethernet_DTSRC 22-12 [1], 31-116 [2]
Ethernet_MACCAMADDR 22-73 [1], 31-61 [2]
Ethernet_MACCAMCTRL0 22-73 [1], 31-53 [2]
Ethernet_MACCAMCTRL1 22-73 [1], 31-62 [2]
Ethernet_MACCAMDATA 22-73 [1], 31-61 [2]
Ethernet_MACCTRL 22-73 [1], 31-52 [2]
Ethernet_MACMERRCNT 22-73 [1], 31-63 [2]
Ethernet_MACPSECNT 22-73 [1], 31-64 [2]
Ethernet_MACRPSECNT 22-73 [1], 31-65 [2]
Ethernet_MACRX0IMR 22-72 [1], 31-76 [2]
Ethernet_MACRX0ISR 22-72 [1], 31-74 [2]
Ethernet_MACRX0SRC 22-11 [1], 31-116 [2]
Ethernet_MACRX1IMR 22-72 [1], 31-80 [2]
Ethernet_MACRX1ISR 22-72 [1], 31-78 [2]
Ethernet_MACRX1SRC 22-12 [1], 31-116 [2]
Ethernet_MACRXCTRL 22-73 [1], 31-57 [2]
Ethernet_MACRXSTAT 22-73 [1], 31-58 [2]
Ethernet_MACSMCTRL 22-73 [1], 31-60 [2]
Ethernet_MACSMDATA 22-73 [1], 31-59 [2]
Ethernet_MACTX0IMR 22-73 [1]
Ethernet_MACTX0ISR 22-73 [1]
Ethernet_MACTX0SRC 22-11 [1], 31-116 [2]
Ethernet_MACTX1IMR 22-72 [1]
Ethernet_MACTX1ISR 22-72 [1]
Ethernet_MACTX1SRC 22-11 [1], 31-116 [2]
Ethernet_MACTXCTRL 22-73 [1], 31-54 [2]
Ethernet_MACTXSTAT 22-73 [1], 31-55 [2]
Ethernet_RBCBL 22-72 [1], 31-100 [2]
Ethernet_RBCC 22-72 [1], 31-99 [2]
Ethernet_RBFPCNT 22-72 [1], 31-103 [2]
Ethernet_RBFPM 22-72 [1], 31-101 [2]
Ethernet_RBFPTH 22-72 [1], 31-102 [2]

Ethernet_RBSRC0 22-12 [1], 31-116 [2]
Ethernet_RBSRC1 22-12 [1], 31-116 [2]
Ethernet_TBCC 22-72 [1], 31-105 [2]
Ethernet_TBCPR 22-72 [1], 31-106 [2]
Ethernet_TBISR 22-72 [1], 31-104 [2]
Ethernet_TBSRC 22-12 [1], 31-116 [2]
EXEVT 2-34 [1], 22-79 [1]

F

FCX 2-23 [1], 22-81 [1]
FDR 4-7 [1], 22-11 [1]
FMR 15-45 [1], 22-12 [1]
FPU_SRC 15-26 [1], 22-12 [1]
FSR 4-6 [1], 22-11 [1]

G

GPTU module registers 28-22 [2]
GPTU_CLC 22-19 [1], 28-54 [2]
GPTU_ID 22-19 [1]
GPTU_OSEL 22-19 [1], 28-47 [2]
GPTU_OUT 22-19 [1], 28-48 [2]
GPTU_SRC0 22-21 [1], 28-61 [2]
GPTU_SRC1 22-21 [1], 28-61 [2]
GPTU_SRC2 22-21 [1], 28-61 [2]
GPTU_SRC3 22-20 [1], 28-61 [2]
GPTU_SRC4 22-20 [1], 28-61 [2]
GPTU_SRC5 22-20 [1], 28-61 [2]
GPTU_SRC6 22-20 [1], 28-61 [2]
GPTU_SRC7 22-20 [1], 28-61 [2]
GPTU_SRSEL 22-20 [1], 28-49 [2]
GPTU_T012RUN 22-20 [1], 28-41 [2]
GPTU_T01IRS 22-19 [1]
GPTU_T01IRST 28-24 [2]
GPTU_T01OTS 22-19 [1], 28-27 [2]
GPTU_T0CBA 22-19 [1], 28-29 [2]
GPTU_T0DCBA 22-19 [1], 28-29 [2]
GPTU_T0RCBA 22-20 [1], 28-30 [2]
GPTU_T0RDCBA 22-19 [1], 28-30 [2]
GPTU_T1CBA 22-20 [1], 28-31 [2]
GPTU_T1DCBA 22-20 [1], 28-31 [2]
GPTU_T1RCBA 22-20 [1], 28-32 [2]
GPTU_T1RDCBA 22-20 [1], 28-31 [2]
GPTU_T2 22-20 [1], 28-45 [2]
GPTU_T2AIS 22-19 [1], 28-33 [2]
GPTU_T2BIS 22-19 [1], 28-35 [2]
GPTU_T2CON 22-19 [1], 28-38 [2]
GPTU_T2ES 22-19 [1], 28-36 [2]
GPTU_T2RC0 22-20 [1], 28-46 [2]
GPTU_T2RC1 22-20 [1], 28-46 [2]
GPTU_T2RCCON 22-19 [1], 28-43 [2]

I

ICR 2-27 [1], 15-8 [1], 22-80 [1]
IGCR0 15-46 [1], 22-12 [1]
IGCR1 15-49 [1], 22-12 [1]
IIC module registers 25-12 [2]
IIC_BUSCON 22-64 [1], 25-23 [2]
IIC_CLC 22-64 [1], 25-28 [2]
IIC_ID 22-64 [1]
IIC_PISEL 22-64 [1], 25-13 [2]
IIC_RTB 22-64 [1], 25-25 [2]
IIC_SYSCON 22-64 [1], 25-14 [2]
IIC_WHB SYSCON 22-64 [1]
IIC_WHBSYSCON 25-20 [2]
IIC_XP0SRC 22-64 [1], 25-32 [2]
IIC_XP1SRC 22-64 [1], 25-32 [2]
IIC_XP2SRC 22-64 [1], 25-32 [2]
ISP 2-26 [1], 22-80 [1]

L

LBCU module registers 18-6 [1]
LBCU_ID 22-92 [1]
LBCU_LEADDR 18-7 [1], 22-92 [1]
LBCU_LEATT 18-8 [1], 22-92 [1]
LBCU_LEDATH 18-10 [1], 22-92 [1]
LBCU_LEDATL 18-10 [1], 22-92 [1]
LBCU_SRC 18-11 [1], 22-92 [1]
LCX 2-25 [1], 22-81 [1]
LFI module registers 18-17 [1]
LFI_CON 18-17 [1], 22-93 [1]
LFI_ID 22-93 [1]

M

MANID 4-16 [1], 22-11 [1]
MCHK_ID 22-71 [1]
MCHK_IR 17-92 [1], 22-71 [1]

MCHK_RR 17-93 [1], 22-71 [1]
MCHK_WR 17-94 [1], 22-71 [1]
Memory protection system registers 12-4 [1]
MLI module registers 27-59 [2]
MLI0_AER 22-67 [1], 27-95 [2]
MLI0_ARR 22-67 [1], 27-96 [2]
MLI0_FDR 22-65 [1], 27-107 [2]
MLI0_GINTR 22-67 [1], 27-94 [2]
MLI0_ID 22-65 [1]
MLI0_OICR 22-67 [1], 27-81 [2]
MLI0_RADDR 27-79 [2]
MLI0_RADRR 22-67 [1]
MLI0_RCR 22-66 [1], 27-74 [2]
MLI0_RDATAR 22-67 [1], 27-79 [2]
MLI0_RIER 22-67 [1], 27-90 [2]
MLI0_RINPR 22-67 [1], 27-93 [2]
MLI0_RISR 22-67 [1], 27-92 [2]
MLI0_RP0BAR 22-66 [1]
MLI0_RP0STATR 22-66 [1]
MLI0_RP1BAR 22-66 [1]
MLI0_RP1STATR 22-66 [1]
MLI0_RP2BAR 22-66 [1]
MLI0_RP2STATR 22-67 [1]
MLI0_RP3BAR 22-66 [1]
MLI0_RP3STATR 22-67 [1]
MLI0_RPxBAR 27-77 [2]
MLI0_RPxSTATR 27-78 [2]
MLI0_SCR 22-67 [1], 27-80 [2]
MLI0_TAOFF 22-68 [1]
MLI0_TCBAR 22-66 [1], 27-73 [2]
MLI0_TCMDR 22-65 [1], 27-67 [2]
MLI0_TCR 22-65 [1], 27-62 [2]
MLI0_TDRAR 22-66 [1], 27-72 [2]
MLI0_TIER 22-67 [1], 27-86 [2]
MLI0_TINPR 22-67 [1], 27-88 [2]
MLI0_TISR 22-67 [1], 27-87 [2]
MLI0_TP0AOFR 22-65 [1]
MLI0_TP0BAR 22-66 [1]
MLI0_TP0DATAR 22-65 [1]
MLI0_TP0STATR 22-65 [1]
MLI0_TP1AOFR 22-65 [1]
MLI0_TP1BAR 22-66 [1]
MLI0_TP1DATAR 22-66 [1]
MLI0_TP1STATR 22-65 [1]
MLI0_TP2AOFR 22-65 [1]
MLI0_TP2BAR 22-66 [1]
MLI0_TP2DATAR 22-66 [1]
MLI0_TP2STATR 22-65 [1]
MLI0_TP3AOFR 22-65 [1]
MLI0_TP3BAR 22-66 [1]
MLI0_TP3DATAR 22-66 [1]
MLI0_TP3STATR 22-65 [1]
MLI0_TPxAOFR 27-71 [2]
MLI0_TPxBAR 27-72 [2]
MLI0_TPxDATAR 27-71 [2]
MLI0_TPxSTATR 27-66 [2]
MLI0_TRSTATR 22-65 [1], 27-69 [2]
MLI0_TSTATR 22-65 [1], 27-64 [2]
MLI1_AER 22-70 [1], 27-95 [2]
MLI1_ARR 22-70 [1], 27-96 [2]
MLI1_FDR 22-68 [1], 27-107 [2]
MLI1_GINTR 22-70 [1], 27-94 [2]
MLI1_ID 22-68 [1]
MLI1_OICR 22-70 [1], 27-81 [2]
MLI1_RADDR 27-79 [2]
MLI1_RADRR 22-70 [1]
MLI1_RCR 22-69 [1], 27-74 [2]
MLI1_RDATAR 22-70 [1], 27-79 [2]
MLI1_RIER 22-70 [1], 27-90 [2]
MLI1_RINPR 22-70 [1], 27-93 [2]
MLI1_RISR 22-70 [1], 27-92 [2]
MLI1_RP0BAR 22-69 [1]
MLI1_RP0STATR 22-69 [1]
MLI1_RP1BAR 22-69 [1]
MLI1_RP1STATR 22-69 [1]
MLI1_RP2BAR 22-69 [1]
MLI1_RP2STATR 22-70 [1]
MLI1_RP3BAR 22-69 [1]
MLI1_RP3STATR 22-70 [1]
MLI1_RPxBAR 27-77 [2]
MLI1_RPxSTATR 27-78 [2]
MLI1_SCR 22-70 [1], 27-80 [2]
MLI1_TCBAR 22-69 [1], 27-73 [2]
MLI1_TCMDR 22-68 [1], 27-67 [2]
MLI1_TCR 22-68 [1], 27-62 [2]

MLI1_TDRAR 22-69 [1], 27-72 [2]
MLI1_TIER 22-70 [1], 27-86 [2]
MLI1_TINPR 22-70 [1], 27-88 [2]
MLI1_TISR 22-70 [1], 27-87 [2]
MLI1_TP0AOFR 22-68 [1]
MLI1_TP0BAR 22-69 [1]
MLI1_TP0DATAR 22-68 [1]
MLI1_TP0STATR 22-68 [1]
MLI1_TP1AOFR 22-68 [1]
MLI1_TP1BAR 22-69 [1]
MLI1_TP1DATAR 22-69 [1]
MLI1_TP1STATR 22-68 [1]
MLI1_TP2AOFR 22-68 [1]
MLI1_TP2BAR 22-69 [1]
MLI1_TP2DATAR 22-69 [1]
MLI1_TP2STATR 22-68 [1]
MLI1_TP3BAR 22-69 [1]
MLI1_TP3DATAR 22-69 [1]
MLI1_TP3STATR 22-68 [1]
MLI1_TPxAOFR 27-71 [2]
MLI1_TPxBAR 27-72 [2]
MLI1_TPxDATAR 27-71 [2]
MLI1_TPxSTATR 27-66 [2]
MLI1_TRSTATR 22-68 [1], 27-69 [2]
MLI1_TSTATR 22-68 [1], 27-64 [2]
MMU module registers 10-13 [1]
MMU_ASI 10-15 [1], 22-75 [1]
MMU_CON 10-13 [1], 22-75 [1]
MMU_ID 22-75 [1]
MMU_TFA 10-18 [1], 22-76 [1]
MMU_TPA 10-16 [1], 22-75 [1]
MMU_TPX 10-18 [1], 22-76 [1]
MMU_TVA 10-15 [1], 22-75 [1]

N

NMISR 16-20 [1], 22-10 [1]

O

OSC_CON 3-8 [1], 22-10 [1]

P

P0_ALTSEL0 13-11 [1], 22-22 [1]
P0_ALTSEL1 13-11 [1], 22-22 [1]
P0_DIR 13-7 [1], 22-22 [1]
P0_IN 13-6 [1], 22-22 [1]
P0_OD 13-8 [1], 22-22 [1]
P0_OUT 13-5 [1], 22-22 [1]
P0_PUDEN 13-10 [1], 22-22 [1]
P0_PUDSEL 13-9 [1], 22-22 [1]
P1_ALTSEL0 13-11 [1], 22-23 [1]
P1_ALTSEL1 13-11 [1], 22-23 [1]
P1_DIR 13-7 [1], 22-23 [1]
P1_IN 13-6 [1], 22-23 [1]
P1_OD 13-8 [1], 22-23 [1]
P1_OUT 13-5 [1], 22-23 [1]
P1_PUDEN 13-10 [1], 22-23 [1]
P1_PUDSEL 13-9 [1], 22-23 [1]
P2_ALTSEL0 13-11 [1], 22-24 [1]
P2_ALTSEL1 13-11 [1], 22-24 [1]
P2_DIR 13-7 [1], 22-24 [1]
P2_IN 13-6 [1], 22-24 [1]
P2_OD 13-8 [1], 22-24 [1]
P2_OUT 13-5 [1], 22-24 [1]
P2_PUDEN 13-10 [1], 22-24 [1]
P2_PUDSEL 13-9 [1], 22-24 [1]
P3_ALTSEL0 13-11 [1], 22-25 [1]
P3_ALTSEL1 13-11 [1], 22-25 [1]
P3_DIR 13-7 [1], 22-25 [1]
P3_IN 13-6 [1], 22-25 [1]
P3_OD 13-8 [1], 22-25 [1]
P3_OUT 13-5 [1], 22-25 [1]
P3_PUDEN 13-10 [1], 22-25 [1]
P3_PUDSEL 13-9 [1], 22-25 [1]
P4_ALTSEL0 13-11 [1], 22-26 [1]
P4_ALTSEL1 13-11 [1], 22-26 [1]
P4_DIR 13-7 [1], 22-26 [1]
P4_IN 13-6 [1], 22-26 [1]
P4_OD 13-8 [1], 22-26 [1]
P4_OUT 13-5 [1], 22-26 [1]
P4_PUDEN 13-10 [1], 22-26 [1]
P4_PUDSEL 13-9 [1], 22-26 [1]
PC 2-17 [1], 22-80 [1]
PCX 2-24 [1]
PCXI 2-22 [1], 22-80 [1]
PLL_CLC 3-14 [1], 22-11 [1]
PMG_CON 6-4 [1]

PMG_CSR 6-5 [1], 22-10 [1]
PMI module registers 8-6 [1]
PMI_CON0 8-7 [1], 22-91 [1]
PMI_CON1 8-8 [1], 22-91 [1]
PMI_CON2 8-9 [1], 22-91 [1]
PMI_ID 22-91 [1]
Ports registers 13-4 [1]
Power management registers 6-3 [1]
PSW 2-18 [1], 12-7 [1], 22-80 [1]

R

Reset registers 5-2 [1]
RST_REQ 5-5 [1], 22-10 [1]
RST_SR 5-3 [1], 22-10 [1]
RTID 4-18 [1], 22-11 [1]

S

SBCU module registers 18-27 [1]
SBCU_CON 18-28 [1], 22-14 [1]
SBCU_DBADR1 18-37 [1], 22-14 [1]
SBCU_DBADR2 18-37 [1], 22-14 [1]
SBCU_DBADRT 18-35 [1], 22-14 [1]
SBCU_DBBOS 18-36 [1], 22-14 [1]
SBCU_DBBOST 18-33 [1], 22-15 [1]
SBCU_DBCNTL 18-39 [1], 22-14 [1]
SBCU_DBGNTT 18-35 [1], 22-14 [1]
SBCU_DBGRNT 18-38 [1], 22-14 [1]
SBCU_EADD 18-32 [1], 22-14 [1]
SBCU_ECON 18-31 [1], 22-14 [1]
SBCU_EDAT 18-32 [1], 22-14 [1]
SBCU_ID 22-14 [1]
SBCU_SRC 18-42 [1], 22-15 [1]
SCU module registers 4-19 [1]
SCU_CON 4-12 [1], 22-11 [1]
SCU_ID 22-10 [1]
SCU_PETCR 4-4 [1], 22-11 [1]
SCU_PETSR 4-4 [1], 22-11 [1]
SCU_SCLIR 5-15 [1], 22-11 [1]
SCU_STAT 4-15 [1], 22-11 [1]
SETA 11-13 [1], 22-89 [1]
SRAR00 22-88 [1]
SRAR01 22-88 [1]
SRAR02 22-88 [1]
SRAR03 22-88 [1]
SRAR04 22-88 [1]
SRAR05 22-88 [1]
SRAR06 22-88 [1]
SRAR07 22-88 [1]
SRAR08 22-88 [1]
SRAR09 22-88 [1]
SRAR10 22-88 [1]
SRAR11 22-88 [1]
SRAR12 22-88 [1]
SRAR13 22-88 [1]
SRAR14 22-88 [1]
SRAR15 22-89 [1]
SRAR16 22-89 [1]
SRAR17 22-89 [1]
SRAR18 22-89 [1]
SRAR19 22-89 [1]
SRAR20 22-89 [1]
SRAR21 22-89 [1]
SRAR22 22-89 [1]
SRAR23 22-89 [1]
SRARn 11-10 [1]
SSC module registers 24-28 [2]
SSC0_BR 22-56 [1], 24-38 [2]
SSC0_CLC 22-56 [1], 24-50 [2]
SSC0_CON 22-56 [1], 24-31 [2]
SSC0_EFM 22-56 [1], 24-34 [2]
SSC0_ESRC 22-57 [1], 24-74 [2]
SSC0_FDR 22-56 [1], 24-51 [2]
SSC0_FSTAT 22-56 [1], 24-44 [2]
SSC0_ID 22-56 [1]
SSC0_PISEL 22-56 [1], 24-29 [2], 24-54 [2]
SSC0_RB 22-56 [1], 24-39 [2]
SSC0_RSRC 22-57 [1], 24-74 [2]
SSC0_RXFCON 22-56 [1], 24-40 [2]
SSC0_SSOC 22-56 [1], 24-36 [2]
SSC0_SSOTC 22-56 [1], 24-37 [2]
SSC0_STAT 22-56 [1], 24-33 [2]
SSC0_TB 22-56 [1], 24-39 [2]
SSC0_TSRC 22-57 [1], 24-74 [2]
SSC0_TXFCON 22-56 [1], 24-42 [2]
SSC1_BR 22-57 [1], 24-38 [2]

SSC1_CLC 22-57 [1], 24-50 [2]
SSC1_CON 22-57 [1], 24-31 [2]
SSC1_EFM 22-58 [1], 24-34 [2]
SSC1_ESRC 22-58 [1], 24-74 [2]
SSC1_FDR 22-57 [1], 24-51 [2]
SSC1_FSTAT 22-58 [1], 24-44 [2]
SSC1_ID 22-57 [1]
SSC1_PISEL 22-57 [1], 24-29 [2], 24-55 [2]
SSC1_RB 22-57 [1], 24-39 [2]
SSC1_RSRC 22-58 [1], 24-74 [2]
SSC1_RXFCON 22-58 [1], 24-40 [2]
SSC1_SSOC 22-57 [1], 24-36 [2]
SSC1_SSOTC 22-57 [1], 24-37 [2]
SSC1_STAT 22-57 [1], 24-33 [2]
SSC1_TB 22-57 [1], 24-39 [2]
SSC1_TSRC 22-58 [1], 24-74 [2]
SSC1_TXFCON 22-58 [1], 24-42 [2]
STM module registers 19-6 [1]
STM_CAP 19-8 [1], 22-16 [1]
STM_CLC 19-15 [1], 22-16 [1]
STM_CMCON 19-10 [1], 22-16 [1]
STM_CMP0 19-9 [1], 22-16 [1]
STM_CMP1 19-9 [1], 22-16 [1]
STM_ICR 19-12 [1], 22-16 [1]
STM_ID 22-16 [1]
STM_ISRR 19-14 [1], 22-16 [1]
STM_SRC0 19-18 [1], 22-16 [1]
STM_SRC1 19-18 [1], 22-16 [1]
STM_TIM0 19-7 [1], 22-16 [1]
STM_TIM1 19-7 [1], 22-16 [1]
STM_TIM2 19-7 [1], 22-16 [1]
STM_TIM3 19-7 [1], 22-16 [1]
STM_TIM4 19-8 [1], 22-16 [1]
STM_TIM5 19-8 [1], 22-16 [1]
STM_TIM6 19-8 [1], 22-16 [1]
SWEVT 2-36 [1], 22-79 [1]
SYSCON 2-31 [1], 22-80 [1]

T

TR0EVT 2-37 [1], 22-79 [1]
TR1EVT 2-37 [1], 22-79 [1]

U

USB module registers 26-21 [2]
USB_CLC 22-52 [1], 26-68 [2]
USB_CNFR 22-53 [1], 26-34 [2]
USB_CPLPR 22-53 [1], 26-48 [2]
USB_DATA16 22-53 [1], 26-50 [2]
USB_DATA32 22-53 [1], 26-49 [2]
USB_DATA8 22-53 [1], 26-50 [2]
USB_DCR 22-53 [1], 26-27 [2]
USB_DIER 22-54 [1], 26-54 [2]
USB_DINP 22-54 [1], 26-53 [2]
USB_DIRR 22-54 [1], 26-56 [2]
USB_DIRST 22-54 [1], 26-58 [2]
USB_DSR 22-53 [1], 26-29 [2]
USB_EPBC0 22-50 [1]
USB_EPBC1 22-50 [1]
USB_EPBC10 22-52 [1]
USB_EPBC2 22-50 [1]
USB_EPBC3 22-51 [1]
USB_EPBC4 22-51 [1]
USB_EPBC5 22-51 [1]
USB_EPBC6 22-51 [1]
USB_EPBC7 22-52 [1]
USB_EPBC8 22-52 [1]
USB_EPBC9 22-52 [1]
USB_EPBCn 26-39 [2]
USB_EPCP0 22-50 [1]
USB_EPCP1 22-50 [1]
USB_EPCP10 22-52 [1]
USB_EPCP2 22-50 [1]
USB_EPCP3 22-50 [1]
USB_EPCP4 22-51 [1]
USB_EPCP5 22-51 [1]
USB_EPCP6 22-51 [1]
USB_EPCP7 22-51 [1]
USB_EPCP8 22-52 [1]
USB_EPCP9 22-52 [1]
USB_EPCPn 26-38 [2]
USB_EPDIR 22-53 [1], 26-40 [2]
USB_EPDSR 22-53 [1], 26-41 [2]
USB_EPIC0 22-50 [1]
USB_EPIC1 22-50 [1]

USB_EPIC10 22-52 [1]
USB_EPIC2 22-50 [1]
USB_EPIC3 22-51 [1]
USB_EPIC4 22-51 [1]
USB_EPIC5 22-51 [1]
USB_EPIC6 22-51 [1]
USB_EPIC7 22-52 [1]
USB_EPIC8 22-52 [1]
USB_EPIC9 22-52 [1]
USB_EPICn 26-60 [2]
USB_EPIR0 22-54 [1], 26-63 [2]
USB_EPIR1 22-54 [1], 26-63 [2]
USB_EPIR2 22-54 [1], 26-63 [2]
USB_EPIR3 22-54 [1], 26-63 [2]
USB_EPIRST0 22-54 [1], 26-64 [2]
USB_EPIRST1 22-54 [1], 26-64 [2]
USB_EPIRST2 22-54 [1], 26-64 [2]
USB_EPIRST3 22-54 [1], 26-64 [2]
USB_EPSSR 22-53 [1], 26-33 [2]
USB_EPSTL 22-53 [1], 26-32 [2]
USB_EPUP0 22-50 [1]
USB_EPUP1 22-50 [1]
USB_EPUP10 22-52 [1]
USB_EPUP2 22-50 [1]
USB_EPUP3 22-50 [1]
USB_EPUP4 22-51 [1]
USB_EPUP5 22-51 [1]
USB_EPUP6 22-51 [1]
USB_EPUP7 22-51 [1]
USB_EPUP8 22-52 [1]
USB_EPUP9 22-52 [1]
USB_EPUPn 26-37 [2]
USB_EPVLD 22-53 [1], 26-43 [2]
USB_EVSR 22-53 [1], 26-44 [2]
USB_FCON 22-53 [1], 26-45 [2]
USB_FNR 22-53 [1], 26-34 [2]
USB_ID 22-53 [1]
USB_PISEL 22-52 [1], 26-26 [2]
USB_SRC0 22-55 [1], 26-74 [2]
USB_SRC1 22-55 [1], 26-74 [2]
USB_SRC2 22-55 [1], 26-74 [2]
USB_SRC3 22-55 [1], 26-74 [2]
USB_SRC4 22-55 [1], 26-74 [2]
USB_SRC5 22-55 [1], 26-74 [2]
USB_SRC6 22-55 [1], 26-74 [2]
USB_SRC7 22-55 [1], 26-74 [2]
USB_SUTH 22-50 [1], 26-35 [2]
USB_SUTL 22-50 [1], 26-35 [2]
USB_ZLPEN 22-54 [1], 26-42 [2]
USB_ZLPSR 22-54 [1], 26-42 [2]

W

WDT module registers 20-28 [1]
WDT_CON0 20-29 [1], 22-10 [1]
WDT_CON1 20-31 [1], 22-10 [1]
WDT_SR 20-32 [1], 22-10 [1]

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG